# POLITECNICO DI TORINO

## MASTER's Degree in Data Science and Engineering


## MASTER's Degree Thesis

# Real-World Industrial Deployment of Vision–Based Robotic Manipulation in the Beverage Industry

**Supervisors**

**Prof. Giuseppe Bruno AVERTA**

**Dr. Davide BUOSO**

**Candidate**

**Arash DANESHVAR**

**March 2026**

## Abstract

This research presents the development and industrial deployment of a vision-guided robotic manipulation system for automated handling tasks in the beverage industry. Reliable object perception and real-time robot control remain challenging in production environments due to variations in lighting, object appearance, workspace conditions, and vision-robot communication. To address these challenges, the proposed system integrates deep learning–based visual perception with collaborative robot manipulation.

The system consists of two main components: robotic perception and robotic manipulation. The perception module employs object detection models to identify different types of cups and estimate their 3D positions using RGB-D image data. A domain-specific dataset collected in a real industrial workspace was used to train and evaluate multiple variants of YOLOv8 models, selected for their balance between detection accuracy and real-time performance on an edge device. To enhance robustness under environmental variability and to verify delivery spot occupancy, a hybrid perception strategy combining deep learning–based detection and depth-difference subtraction was implemented.

For robotic manipulation, the estimated cup positions are transformed from the camera coordinate frame to the robot coordinate frame through an Eye-to-Hand calibration procedure. The resulting target positions are transmitted to the robot controller, which executes the pick-and-place motion using ABB's Externally Guided Motion (EGM) interface.

The system was fully deployed in a real industrial production environment. Experimental evaluation shows that the perception model achieves **94.1% mAP@50–95** with an inference time of **5.4 ms**, while the robotic system reaches a **93.33% pick-and-place success rate** during production tests.

These results demonstrate the practical feasibility and effectiveness of deploying vision-guided robotic manipulation systems for real-world industrial automation.

ACKNOWLEDGMENTS

First of all, I would like to thank my family for their constant love and support throughout this journey. Their encouragement helped me continue during difficult times. I am especially grateful to my sister and her family for always being there for me and supporting me in many ways during this path.

I would also like to thank my supervisors at both the university and the company for their guidance and support throughout this research. Their advice and feedback helped me a lot and played an important role in my academic and personal development.

I am also thankful to my friends and colleagues who shared this journey with me. Your support, conversations, and encouragement made this experience much more meaningful and enjoyable. I am grateful to everyone who accompanied me along this path.

This thesis was written during a difficult time for both my fellow countrymen and me. While I was working on this research, many Iranians were fighting for freedom, and many lost their lives along the way. The courage and sacrifice of those who gave their lives during this time will always remain in my memory.

**This thesis is dedicated to all those who gave their lives for the freedom of Iran,**

**especially the victims of 8 and 9 January 2026.**

**Your memory will never be forgotten.**

# Table of Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| AI | Artificial Intelligence. |
| CLIP | Contrastive Language–Image Pretraining. |
| CNN | Convolutional Neural Network. |
| Cobot | Collaborative Robot. |
| CUDA | Compute Unified Device Architecture. |
| DDS | Data Distribution Service. |
| DETR | Detection Transformer. |
| DINO | Self-Distillation with No Labels. |
| EGM | Externally Guided Motion. |
| FPN | Feature Pyramid Network. |
| GLIP | Grounded Language-Image Pretraining. |
| GPU | Graphics Processing Unit. |
| HOG | Histogram of Oriented Gradients. |
| RAPID | Robot Application Programming Interactive Dialogue. |
| R-CNN | Region-based Convolutional Neural Network. |
| RGB-D | Red Green Blue with Depth. |
| ROI | Region of Interest. |
| ROS | Robot Operating System. |
| RPN | Region Proposal Network. |
| RT-DETR | Real-Time Detection Transformer. |
| SPP | Spatial Pyramid Pooling. |
| SSD | Single Shot MultiBox Detector. |
| TCP | Tool Center Point. |

| | |
|---|---|
| UDP | User Datagram Protocol. |
| ViLD | Vision-Language Model for Open-Vocabulary Detection. |
| YOLO | You Only Look Once. |

# Chapter 1

# Introduction

Recent advances in deep learning have significantly improved the performance of vision-based object detection systems [1, 2, 3, 4]. These developments have opened new opportunities for their application in robotic manipulation tasks. However, while such methods achieve high accuracy in controlled laboratory environments, deploying them reliably in real industrial settings remains a challenging task. This thesis investigates the development and industrial deployment of a vision-based robotic manipulation system designed to operate under real production conditions.

In industrial production environments, detecting the presence of an object is commonly performed using sensor-based approaches. Although this approach is effective for specific tasks, but sensor-based solutions lack scalability and flexibility, particularly when production systems expand. Each additional detection part requires additional physical sensors, wiring, calibration, and maintenance.

At the same time, collaborative robots (cobots) have become increasingly common in manufacturing [5], specially for tasks like human-robot interaction due to their flexibility and ability to operate safely alongside humans. When combined with vision systems, cobots can perform manipulation tasks such as pick-and-place operations [6]. However, integrating perception with robotic manipulation requires a unified and reliable system. Achieving this integration involves several key components, including camera-to-robot calibration, embedded computation, accurate 3D localization, and real-time motion control.

## 1.1 Problem Statement

The primary motivation for this research arises from an industrial need within the company and the opportunity to replace conventional approaches with a more scalable and intelligent solution. Two major technical challenges shaped the development of this work.

First, the existing delivery system relied on photoelectric sensors to verify whether a cup was present at the target delivery spot. Replacing the sensor-based approach with a vision-based system offers several advantages, including improved scalability, reduced hardware dependency, and the flexibility to extend the same system to future

tasks. The developed vision framework is not limited to cup detection but can be adapted to new object detection scenarios and additional robotic manipulation tasks, making it more versatile than conventional sensor-based solutions. However, deploying a reliable vision-based system introduces additional challenges, including varying lighting conditions, different background surfaces, embedded hardware constraints, and real-time performance requirements.

Second, the manipulation system operates using a cobot, creating the opportunity for collaborative manipulation. To take advantage of collaborative robotics, the system must enable reliable communication between external perception devices, such as cameras and sensors, and the robot controller. This requires integration between perception, accurate 3D localization, calibration, vision-robot comunication, and real-time motion control.

Beyond these two challenges, implementing the perception and manipulation systems in a real industrial environment brings further practical difficulties. The system must run on embedded hardware with limited computational resources, satisfy with industrial standards, and operate reliably during continuous production.

## 1.2 Research Objectives

The goal of this research is to develop and validate a vision-based robotic manipulation system that can operate reliably in real industrial environments and handle the kinds of conditions that are typically present in production settings.

To better explain the objectives of this research, the detection mechanism that is currently used in the automation system is first examined in detail. This analysis explains how the existing sensor-based solution operates, how it detects the presence of cups at predefined delivery positions, and what practical limitations can appear when the system is used in real production conditions.

After establishing this understanding, the vision-based delivery detection system proposed in this research is introduced. In this approach, the traditional sensor-based mechanism is replaced with a camera-based perception system that monitors the delivery area and provides visual information that can be used to guide and support robotic manipulation. This information allows the system to detect the presence of cups and estimate their spatial position relative to the robot, enabling more flexible and adaptive delivery operations.

Specifically, this work aims to:

- Develop a robust perception system for real-time cup detection on an embedded system
- Enable vision-guided pick-and-place manipulation using a cobot
- Deploy, integrate, and validate the complete system under practical industrial conditions

### Existing Delivery Detection System

In the current automation system, the presence of cups at the delivery positions is detected using a sensor-based mechanism. The overall configuration of this delivery detection setup is shown in Figure 1.1. In this setup, several predefined delivery locations are used so that the system can determine whether a prepared drink has already been placed at a specific position or whether that position is still available.

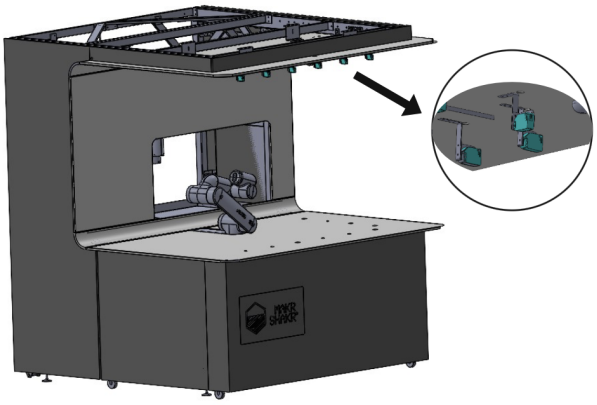


**Figure 1.1:** Existing delivery detection system

To detect the presence of cups at these delivery positions, the system uses retro-reflective photoelectric sensors [7]. In this configuration, a separate sensor is installed at each delivery location so that the system can continuously check whether the position is occupied by a cup or remains empty.

Each photoelectric sensor contains both a light emitter and a receiver inside the same housing. During operation, the sensor emits a beam of light toward a reflector that is placed opposite the sensor. As long as the emitted light reaches the reflector and is reflected back to the receiver, the system interprets this condition as an empty delivery position. When a cup is placed at that location, it blocks the light beam and prevents it from reaching the reflector. Because the light can no longer return to the receiver, the sensor detects this interruption and sends a signal to the control system indicating that the position is now occupied. The operating principle of the retro-reflective photoelectric sensor used in this system is illustrated in Figure 1.2.

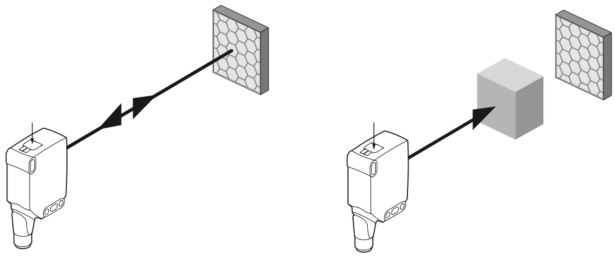


**Figure 1.2:** Retro-reflective photoelectric sensor operating principle *source: [7]*

The output of the sensor is a binary signal that is sent to the robot controller. Based on this signal, the controller determines which delivery position is currently

available for placing a prepared drink. Using this information, the robot then performs the delivery motion and moves the drink to the corresponding predefined location.

Since each delivery position requires its own dedicated sensor, the system depends on several hardware components, including sensors, wiring, calibration procedures, and connections to the control system. Although this approach provides reliable presence detection, it also means that the system grows directly with the number of delivery positions. As the system expands, additional sensors and hardware must be installed, which increases the overall complexity of the system and also leads to higher maintenance effort.

In addition, the performance of photoelectric sensors can be affected by environmental conditions. For example, dust or dirt that accumulates on the reflector surface may reduce the reliability of the signal and can sometimes lead to incorrect detections. Another limitation is that the robot does not receive any visual information from the environment and instead relies only on simple binary signals from the sensors.

These limitations motivate the investigation of alternative sensing approaches that can provide richer information about the environment while also reducing the dependence on multiple hardware components.

### Vision-Based Delivery Detection System

To overcome the limitations of the sensor-based approach described in the previous section, this research proposes a vision-based delivery detection system. Instead of installing a separate photoelectric sensor at every delivery position, the proposed solution uses a camera to observe the entire delivery area and detect whether cups are present at the different delivery locations.

The overall architecture of the proposed vision-based delivery detection system is shown in Figure 1.3. In this configuration, the camera is positioned above the delivery workspace and observes the area from a top-down perspective. From this position, the camera is able to capture visual information from all delivery locations at the same time.

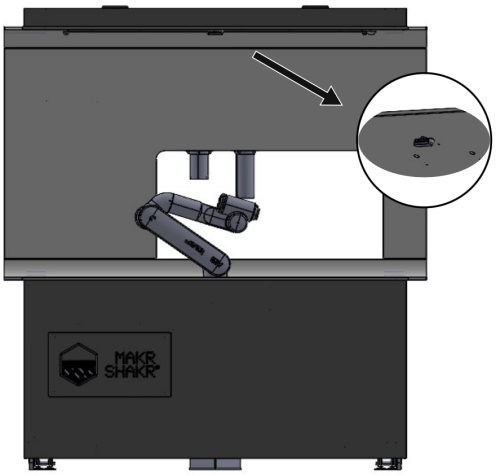



**Figure 1.3:** Vision-based delivery detection system architecture

The images captured by the camera are processed to detect the presence of cups within the workspace. In addition to standard image information, the system can also estimate depth information, which makes it possible to detect the cups or any object using depth-based differences. By using this information, the system can identify which delivery locations are currently occupied and which positions remain available for placing a prepared drink.

Compared with the existing sensor-based solution, the camera-based approach provides much richer information about the environment than simple binary sensor signals. This additional perception capability allows the robot to better understand the current state of the workspace and enables more flexible interaction with the surrounding environment. As a result, the system creates the foundation for vision-guided robotic manipulation.

Using this visual feedback, the robot arm can perform pick-and-place operations while continuously receiving information about the workspace. The robot motion is executed through the Externally Guided Motion (EGM) communication protocol, which allows the robot to interact with the perception system during task execution [8].

## 1.3 Contributions

This work addresses these challenges by developing and deploying a vision-based robotic manipulation system that operates in an industrial production environment. Instead of concentrating only on algorithm optimization, this study examines how such a system can be implemented and used in practice, particularly when addressing the constraints and conditions typical of real-world industrial settings.

The main contributions of this work are:

- Industrial deployment and validation of a vision-based robotic manipulation system within a real production environment at MakrShakr.
- Collection of a domain-specific dataset fine-tuning of the detection model to ensure robust operation under real environmental conditions.
- A computationally efficient hybrid perception framework combining deep learning–based detection with ROI-based depth-difference subtraction, enabling stable real-time inference on embedded platforms such as the NVIDIA Jetson Orin Nano.
- Accurate 3D localization through practical implementation of Eye-to-Hand calibration and geometric transformation between camera and robot coordinate frames.
- Stable real-time motion execution using a Cartesian control strategy integrated via ABB's EGM protocol. [9]

## 1.4 Industrial Collaboration

### 1.4.1 Industrial Partner: Overview

*MakrShakr* is an Italian company focused on the development of bartender robot based on industrial robotic manipulators [10]. The company was founded in Turin in 2014 and focuses on developing automated beverage preparation robots for the hospitality and entertainment industries.

One of the earliest systems developed by MakrShakr was the Bionic Bar, originally created for Royal Caribbean cruise ships. The concept was designed to automate cocktail preparation using robotic arms, allowing customers to select or customize drinks through a digital interface. Once a recipe is submitted, the robotic system autonomously prepares and serves the beverage.

After the initial concept was successfully introduced, MakrShakr expanded its technology into commercial robotic bar systems deployed in various public venues such as airports, entertainment centers, and hospitality environments. Today, MakrShakr installations can be found in multiple locations worldwide, including cruise ships, airports, and large entertainment venues.

The research presented in this thesis was developed in collaboration with MakrShakr and focuses on improving perception and manipulation capabilities in the company's bartending robots.

### 1.4.2 Types of Robots

MakrShakr has developed several bartender robots designed to automate beverage preparation while maintaining high production capacity. These systems combine industrial robotic arms, automated dispensing units, and digital ordering interfaces. Although they share a similar technological concept, each system includes different features to address specific operational needs. These systems aim to bring robotic bartending technology into real-world environments and make automated beverage preparation more accessible. In the following subsections, the company's robot types and their key differences are described.

#### Toni:

The *Toni* platform represents the first system designed for large-scale deployment Figure 1.4(a). It integrates two industrial robotic arms that collaborate during drink preparation tasks such as pouring, shaking, stirring, and serving beverages. The system also incorporates automated bottle dispensing units, ice management systems, and interactive ordering interfaces.

The robotic platform can produce more than 80 drinks per hour and supports a wide variety of drink recipes by combining multiple ingredients. After preparation, one of the robotic arms places the beverage at a designated delivery position where it can be retrieved by the customer.

### Toni Compatto:

In addition to large-scale installations, MakrShakr developed the *Toni Compatto* system Figure 1.4(b), a compact bartender robot designed for smaller venues and easy deployment, often used at events. Unlike the standard Toni platform, this system uses a single robotic arm equipped with a multifunctional end effector that can perform various bartending tasks.

Toni Compatto can prepare both hot and cold beverages, including cocktails, beer, and coffee-based drinks. The system includes automated cup dispensing, bottle storage modules, and beverage preparation units, enabling continuous operation with minimal human intervention.

### Toni Veloce:

To address high-demand environments, MakrShakr developed the *Toni Veloce* robot Figure 1.4(c) . This version focuses on maximizing throughput and efficiency, enabling drink preparation in less than 15 seconds per order. The system includes an optimized beverage management architecture and multiple delivery zones designed to serve a large number of customers simultaneously.

### Toni Compatto Veloce:

The most recent robot developed by MakrShakr is the *Toni Compatto Veloce*. This system as shown in Figure 1.4(d) represents an evolution of the previous Toni Compatto model and is designed to combine compact installation requirements with increased production performance.

The system integrates a cobot arm that operates safely in environments shared with human operators. Unlike the larger Toni and Toni Veloce systems, which are designed as fully automated bar stations, the Compatto Veloce robot is primarily intended for back-of-house operation in professional venues, where the robot assists bartenders by automating repetitive drink preparation tasks.

This robot is capable of producing up to 100 drinks per hour while maintaining high precision in the mixing process. Its compact architecture and optimized dimensions allow installation even in limited spaces, making it suitable for bars, restaurants, and high-traffic hospitality environments.

The introduction of cobots also reflects a significant shift in the company's development strategy. By using ABB GoFa cobot , MakrShakr aims to enable safer human–robot interaction and more flexible system deployment in dynamic hospitality environments.

As a result, cobot arms are expected to play an increasingly important role in the future development of the company's robots.

The work presented in this thesis was carried out on the Toni Compatto Veloce robot. In this system, drinks prepared by the robotic arm are placed at predefined delivery positions where they can be collected by customers or operators. Detecting whether a cup is present at these delivery points is therefore essential for ensuring safe and efficient system operation. In addition, once the operator places a cup on

**(a)** Toni

**(b)** Toni compatto

**(c)** Toni veloce

**(d)** Toni Compatto Veloce

**Figure 1.4:** MakrShakr bartender robot

the table, the robot detects it, determines its position, and picks it up to begin the beverage preparation process. To address these requirements, a vision-based detection and manipulation framework was developed and is presented in this thesis.

### 1.4.3 Benefits of the Industrial Setting

Conducting this research in a real industrial production environment provided several important advantages. First, the proposed system was evaluated under actual operating conditions rather than controlled laboratory setups, which increased the practical relevance and reliability of the results.

Second, access to real production data made it possible to adapt and validate the model under realistic lighting conditions, background variations, and operational constraints. This ensured that the developed solution addressed real-world challenges instead of idealized laboratory scenarios.

In addition, deploying the system in a continuous production workflow allowed the evaluation of its long-term stability, robustness, and real-time performance. Integrating the solution with industrial hardware, embedded platforms, and stan-

dard communication protocols also provided valuable insights into practical system implementation issues.

Beyond its academic contribution, this collaboration also brought clear benefits to the industrial partner. The proposed solution introduced a scalable vision-based capability to the existing production system, reducing reliance on multiple discrete sensors and enabling future functional extensions without major hardware modifications. As a result, the company gained not only a validated technical improvement but also a foundation for further vision-based development.

Overall, conducting the research in an industrial setting increased both the practical impact of the work and its direct value for the collaborating company by bridging the gap between academic research and real-world deployment.

## 1.5 Thesis Structure

This thesis is organized into five main chapters, including the current one. Chapter 2 presents a review of the related literature, covering foundational concepts, relevant tasks and benchmarks, vision–based object detection models, and vision-guide manipulation. Chapter 3 describes the proposed research methodology and explains each component in detail, including the overall architecture of the system, hardware configuration, dataset preparation, vision-based detection framework, camera-to-robot calibration procedure, 3D localization method, and integration of perception with real-time robotic control. Chapter 4 presents the experimental results and industrial validation of the proposed system under real production conditions and the evaluation metrics used. Finally, Chapter 5 concludes the key contributions of the thesis and discusses and suggests potential directions for future work.

# Chapter 2

# Literature Review

This research lies at the intersection of object detection, robotic motion control, and industrial system deployment. This chapter reviews the fundamental concepts and existing research related to object detection methods, vision-guided robotic manipulation, and robot–vision communication frameworks.

## 2.1 Perception

**What is Robotic Perception?** Robotic perception is the ability of a robot to sense and understand its surrounding environment using data from sensors [11]. In many robotic systems, cameras are widely used because they provide rich visual information about the workspace. By analyzing images, robots can detect objects, estimate their positions, and recognize important features of the environment [12].

In recent years, machine learning methods, especially deep learning, have greatly improved vision-based perception systems. Deep neural networks can learn visual patterns directly from data, which allows robots to detect objects and understand scenes more reliably in real-world applications [13]. Among the various tasks performed by robotic perception systems, object detection plays a fundamental role. Object detection enables robots to identify and localize objects within the environment using visual data.

**Object detection models** have evolved significantly over the past decades, moving from traditional methods based on handcrafted features to modern approaches that rely on deep learning for feature extraction [14, 15]. In general, object detection aims to identify objects in an image and localize them by predicting bounding boxes and class labels [16]. Deep learning-based object detectors are commonly grouped into three main categories based on their architecture: two-stage detectors, one-stage detectors, and transformer-based models [14]. Each category has distinct architecture and presents its own trade-offs between accuracy and speed, which makes them suitable for different use cases. Understanding these architectural differences is important when selecting models for real-time industrial applications, where both detection accuracy and computational latency are critical.

The following subsection first presents a brief overview of the evolution of object

detection methods. It then describes the main detector architectures of each category in more detail.

### 2.1.1 Evolution of Object Detection Approaches: Overview

Object detection has evolved from traditional systems based on handcrafted features to modern deep learning architectures. Early approaches relied on manually designed descriptors and sliding-window strategies. Methods such as Viola–Jones [17] and HOG-based detectors [18] achieved practical results, but they required careful feature design and often struggled to generalize to complex scenes.

A major shift occurred with the adoption of deep convolutional neural networks (CNNs) [1], which allowed models to learn features directly from data. This transition was driven by the availability of large-scale datasets and advances in computational hardware, particularly GPU acceleration. As a result, two-stage detection architectures were introduced. Models such as R-CNN [2], Fast R-CNN [19], and Faster R-CNN [3] first generate candidate object regions and then classify them while refining their bounding boxes. These models significantly improved detection accuracy, although they require significant computational resources.

To improve computational efficiency, a different family of object detection architectures, known as one-stage detectors, was introduced. Unlike two-stage detectors, these models perform object localization and classification in a single step without generating intermediate region proposals. Well-known architectures such as YOLO [4] and SSD [20] follow this design principle. By removing the separate region-proposal stage and performing detection in a single forward pass, these models significantly reduce inference time. As a result, one-stage detectors became particularly attractive for real-time applications where fast processing is required. Early versions of these models generally achieved lower detection accuracy compared to two-stage detectors. However, improvements in model architectures and training strategies over time have significantly reduced this gap. Today, one-stage detectors offer a good balance between accuracy and speed, making them widely used in real-time vision systems [21]. Figure 2.1 show the camparison between different models.

| Methods | Trained on | mAP(%) | Test time(sec/img) | Rate(FPS) |
|---|---|---|---|---|
| SS+R-CNN [15] | 07 | 66.0 | 32.84 | 0.03 |
| SS+SPP-net [64] | 07 | 63.1 | 2.3 | 0.44 |
| SS+FRCN [16] | 07+12 | 66.9 | 1.72 | 0.6 |
| SDP+CRC [33] | 07 | 68.9 | 0.47 | 2.1 |
| SS+HyperNet* [101] | 07+12 | 76.3 | 0.20 | 5 |
| MR-CNN&S-CNN [110] | 07+12 | 78.2 | 30 | 0.03 |
| ION [95] | 07+12+S | 79.2 | 1.92 | 0.5 |
| Faster R-CNN(VGG16) [18] | 07+12 | 73.2 | 0.11 | 9.1 |
| Faster R-CNN(ResNet101) [18] | 07+12 | **83.8** | 2.24 | 0.4 |
| YOLO [17] | 07+12 | 63.4 | **0.02** | **45** |
| SSD300 [71] | 07+12 | 74.3 | **0.02** | **46** |
| SSD512 [71] | 07+12 | 76.8 | 0.05 | 19 |
| R-FCN(ResNet101) [65] | 07+12+coco | 83.6 | 0.17 | 5.9 |
| YOLOv2(544*544) [72] | 07+12 | 78.6 | 0.03 | 40 |
| DSSD321(ResNet101) [73] | 07+12 | 78.6 | 0.07 | 13.6 |
| DSOD300 [74] | 07+12+coco | 81.7 | 0.06 | 17.4 |
| PVANET+ [116] | 07+12+coco | **83.8** | 0.05 | 21.7 |
| PVANET+(compress) [116] | 07+12+coco | 82.9 | 0.03 | 31.3 |

**Figure 2.1:** Comparison of testing consumption on VOC 07 test set *source: [21]*

More recently, transformer architectures have introduced a new direction in object detection. Inspired by their success in natural language processing, researchers began

h

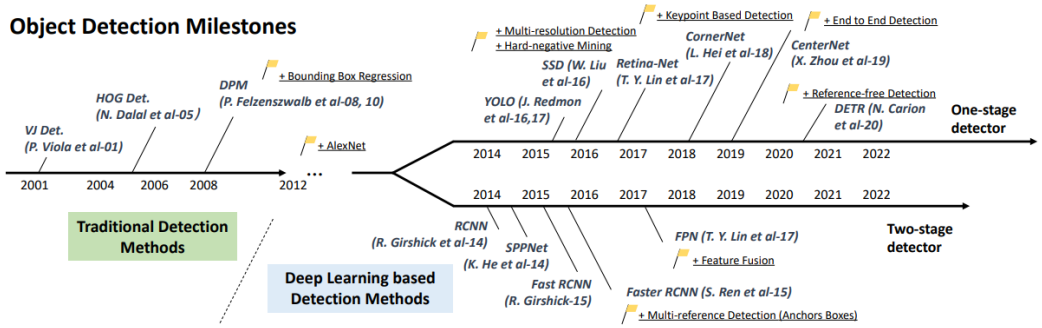



**Figure 2.2:** Evolution of object detection. *Source: [14]* .

adapting transformers to visual tasks. As a result, transformer-based models such as Detection Transformer (DETR) [22] reformulate object detection as a set prediction problem, where the model directly predicts a fixed set of object detections. This formulation removes the need for several components that were traditionally used in detection pipelines, such as anchor boxes and non-maximum suppression. In addition, the attention mechanism in transformers allows the model to capture global contextual relationships within an image more effectively. However, despite their conceptual simplicity and strong contextual reasoning capabilities, these models typically require more computational resources and longer training times than earlier detection architectures.

In parallel, open-vocabulary detection [23] has emerged by integrating vision–language models, allowing detectors to recognize objects beyond predefined training categories. This capability is particularly useful in dynamic environments where object categories may change over time.

Figure 2.2 illustrates the evolution of object detection methods over time and highlights several influential models and publications that have shaped this field. Overall, research in object detection has gradually shifted from traditional approaches based on handcrafted features to data-driven methods built on deep learning.

Over time, these models have become more flexible and capable, improving both detection accuracy and robustness across a wide range of applications. For real-time industrial applications, selecting an appropriate approach requires balancing detection accuracy, latency, computational cost, and environmental robustness.

### 2.1.2 Two-stage Detectors

Two-stage detectors address object detection through two sequential steps: First, they generate a set of candidate regions that may contain objects. Then, these regions are refined by predicting the object category and adjusting the bounding box location. This design usually leads to more accurate localization, but it also introduces additional computational cost compared to single-stage detectors.

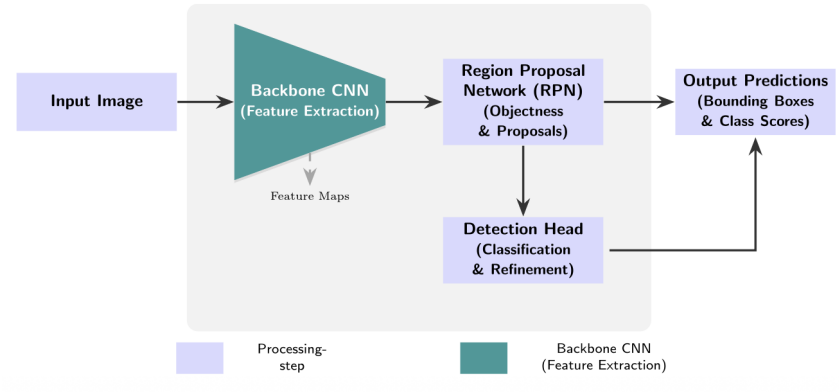



**Figure 2.3:** Representation of a Two-stage object Detector *source: [24]*

**R-CNN** [2] generates candidate object regions using Selective Search. Each region is then processed independently by a convolutional neural network (CNN) to extract visual features. These features are classified using class-specific SVMs, and bounding box regression is applied to refine the object locations. Although R-CNN significantly improves detection accuracy, it is computationally expensive because the CNN must process thousands of region proposals for each image.

**SPPNet** [25] addresses this problem by computing convolutional feature maps only once for the entire image. It introduces a Spatial Pyramid Pooling (SPP) layer that extracts fixed-length features from region proposals of different sizes. This greatly reduces redundant computation, although the training process is still performed in multiple stages.

**Fast R-CNN** [19] further improves efficiency by computing convolutional features once for the entire image and sharing them across all region proposals. It introduces the RoI pooling layer, which converts the features of each proposal into a fixed-size representation. This allows the network to be trained end-to-end and to predict both the object class and the bounding box location in a single model, leading to higher accuracy and faster inference.

**Faster R-CNN** [3] removes the need for external proposal methods by introducing the Region Proposal Network (RPN). The RPN generates object proposals directly from the shared convolutional features using anchor boxes. This makes proposal generation much faster and allows the detection pipeline to be trained in a more integrated way, improving speed while maintaining high accuracy.

**FPN** [26] improves multi-scale object detection by building a feature pyramid from the convolutional features of different network layers. It combines high-level semantic information with low-level spatial details through a top-down pathway and lateral connections. This design greatly improves the detection of small objects and has become a standard component in many modern object detection frameworks.

### 2.1.3 One-stage Detectors

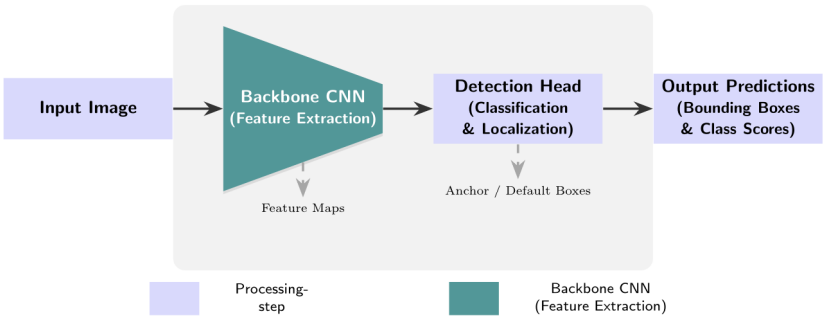
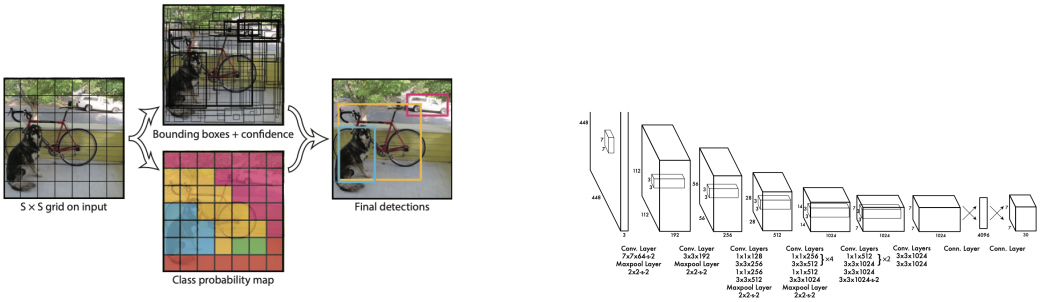



**Figure 2.4:** Representation of a One-stage object Detector *source: [24]*

Unlike two-stage detectors, which process detection in multiple steps, one-stage detectors perform object localization and classification in a single forward pass. By removing the separate region proposal stage, these models reduce computational complexity and achieve faster inference. This simpler design makes them particularly suitable for real-time and industrial applications.

**YOLO (You Only Look Once)** [4] was one of the earliest and most influential one-stage detectors in the deep learning era. It treats object detection as a single regression task, allowing the model to predict bounding boxes and class probabilities directly from the input image. This design enables very fast inference while still maintaining competitive accuracy. Although early versions had difficulty detecting small objects, later versions introduced architectural changes and improved training methods that significantly enhanced performance. Figure 2.5(a) illustrates the grid-based detection mechanism used by YOLO, while Figure 2.5(b) shows the general architecture of the model.




**(a)** Model **(b)** Architecture

**Figure 2.5:** YOLO architecture and model *source: [4]*

**SSD (Single Shot MultiBox Detector)** [20] improves object detection at multiple scales by using feature maps with different resolutions. The model makes predictions on several feature layers, which allows it to detect objects of different sizes more effectively. This design maintains real-time performance while also improving localization accuracy compared to earlier grid-based detectors.

**RetinaNet** [27] addresses a key limitation of early one-stage detectors: the imbalance between foreground and background samples during training. It introduces the Focal Loss function, which reduces the impact of easy negative examples and helps the model focus more on harder samples. This approach improves training stability and significantly reduces the accuracy gap between one-stage and two-stage detectors.

More recent work has explored **anchor-free detectors**, which remove predefined anchor boxes and instead predict object locations using keypoints or center points. For example, **CornerNet** [28] detects objects by predicting the top-left and bottom-right corners of bounding boxes and grouping them to form object detections. Similarly, **CenterNet** [29] represents objects by their center points and directly predicts the corresponding object size and offsets. These approaches simplify the detection pipeline and often improve robustness by avoiding the need for manually designed anchor boxes.

Overall, one-stage detectors have evolved into efficient and accurate models that offer a good balance between detection accuracy and processing speed. Because of this efficiency, they are widely used in embedded systems and real-time industrial applications.

### 2.1.4 Transformer-Based Real-Time Detectors

Following their success in natural language processing, Transformer architectures were rapidly adopted in computer vision. Unlike convolutional neural networks (CNNs), which operate through local receptive fields, Transformers rely on self-attention mechanisms that model global relationships across the entire image. This global reasoning capability addresses some of the contextual limitations of purely convolutional approaches.

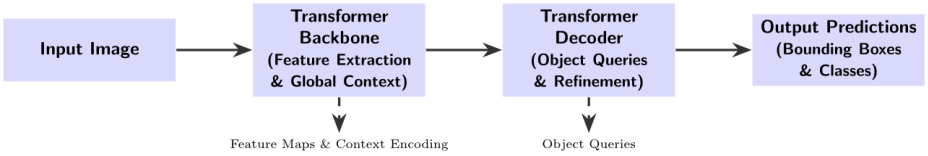



**Figure 2.6:** Representation of a transformer based detector *source: [24]*

In 2020, **DETR** [22] introduced a new formulation of object detection as a direct set prediction problem. By removing traditional components such as anchor boxes and non-maximum suppression, DETR demonstrated that detection could be performed in a fully end-to-end manner. However, the original DETR suffered from slow training convergence and relatively weak performance on small objects.

To overcome these limitations, **Deformable DETR** [30] introduced a more efficient attention mechanism that focuses on sparse, multi-scale sampling around relevant regions. This modification significantly improved convergence speed and detection accuracy.

**RT-DETR** [31] Although Transformer-based detectors simplify the detection pipeline conceptually, they typically require higher computational resources compared to lightweight one-stage CNN-based models. For this reason, recent research has focused on developing more efficient variants, such as real-time Transformer detectors, that aim to balance global reasoning capabilities with practical inference speed.

Overall, Transformer-based models represent an important shift toward unified and learning-driven detection frameworks. However, computational cost and deployment efficiency remain critical considerations, particularly in real-time industrial environments.

### 2.1.5 Open-Vocabulary Detection

Traditional object detection models operate under a closed-set assumption, meaning they can only recognize object categories seen during training. This limitation reduces flexibility in dynamic environments where new object classes may appear.

Open-vocabulary detection addresses this constraint by leveraging vision–language models that align visual features with textual embeddings. Models such as **CLIP** [32] enabled joint image–text representation learning, forming the foundation for subsequent open-vocabulary detection frameworks.

Several approaches have extended this idea to object detection. Methods such as **ViLD** [33] and **Detic** [34] transfer knowledge from vision–language pretraining to region-based detectors, enabling recognition of novel categories. Transformer-based frameworks such as **GLIP** [35] and **Grounding DINO** [36] further integrate cross-modal attention mechanisms to perform grounded detection conditioned on textual prompts.

More recently, efforts have focused on improving efficiency and real-time performance. Architectures such as **YOLO-World** [23] incorporate open-vocabulary capabilities into one-stage detection frameworks, aiming to balance flexibility with inference speed. This direction is particularly relevant for industrial applications where scalability and low latency are essential.

## 2.2 Manipulation

**What is Robotic Manipulation?** Robotic manipulation refers to a robot's ability to physically interact with objects in its environment using its manipulator and end-effector to perform tasks such as grasping, positioning, and pick-and-place operations [37]. Achieving this capability requires the integration of motion modeling, planning algorithms, and control strategies that allow the robot to interact accurately with objects in the workspace.

Traditional robotic manipulation systems typically assume structured environments in which object locations are known in advance. Under such conditions, robots can execute predefined trajectories with high precision. However, these assumptions limit the applicability of such systems in dynamic or uncertain environments where

object positions may vary. To address this limitation, visual perception can be integrated into the manipulation process [38].

The integration of perception with manipulation leads to vision-guided robotic manipulation, in which cameras and computer vision algorithms provide real-time information about object identity and pose to guide robot actions [39]. In such systems, perception modules detect and localize objects within the environment, while the manipulation system transforms this sensory information into physical actions executed by the robot. As a result, vision-guided manipulation requires the coordinated integration of visual perception, calibration, coordinate transformation, and motion control in order to enable reliable interaction with objects.

### 2.2.1 Vision-Guided Robotic Manipulation: Overview

In many industrial production systems, object detection is performed using dedicated sensors such as photoelectric or proximity sensors. These sensors are reliable for simple detection tasks but offer limited flexibility when production lines expand or when new detection points are required. Each additional detection point requires extra hardware, wiring, calibration, and maintenance. Vision-based systems provide a more flexible alternative, since a single camera can monitor multiple locations simultaneously while capturing richer information about the environment [40].

Vision-based techniques are widely used to guide robotic manipulators in industrial environments [38]. In these systems, cameras, image processing algorithms, and robot control frameworks work together to detect objects and estimate their position in the workspace. Different sensing technologies, including stereo vision, structured-light systems, and time-of-flight cameras, can provide the three-dimensional information needed for manipulation tasks [41]. With this information, robots can understand their surroundings and identify target objects before executing manipulation actions.

Recent advances in deep learning have significantly improved vision-based perception systems. CNN-based models are widely used for detecting and localizing objects in complex environments, while learning-based methods such as deep reinforcement learning have also been explored to improve robotic manipulation strategies [42]. These approaches allow robots to identify objects and determine potential grasping locations directly from camera images, enabling more robust manipulation in unstructured or partially unknown environments [43]. As a result, vision-guided manipulation has become an increasingly important approach for flexible automation in modern production systems.

In practical robotic systems, vision-guided manipulation is implemented through a pipeline that connects perception with robot motion. This pipeline typically includes visual sensing, object detection, object localization, camera-to-robot calibration, coordinate transformation, and motion control. Object localization estimates the position and orientation of detected objects relative to the camera frame, which is required for planning the robot motion needed for manipulation [44].

Once the object position has been determined in the robot coordinate system,

motion planning and control algorithms, including feedback control [45] and linear interpolation [46] are used to generate the robot trajectory required for the manipulation task [47]. These algorithms ensure that the robot can reach the target object safely and accurately while respecting kinematic constraints and collision avoidance requirements.

Figure 2.7 illustrates a typical process flow diagram for vision-based autonomous robotic systems. This pipeline typically includes visual sensing, perception processing, localization, calibration, motion planning, and robot control.

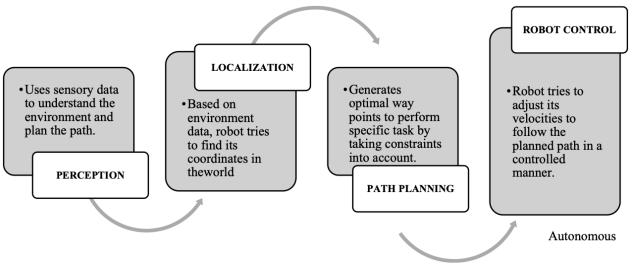
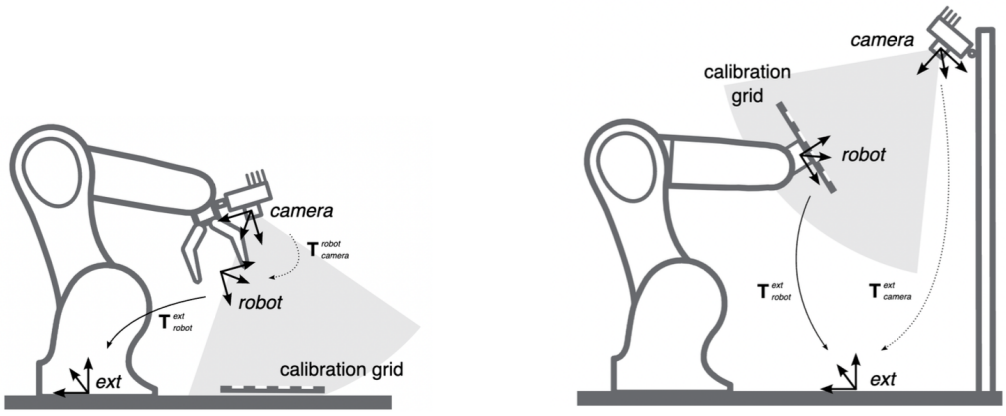



**Figure 2.7:** Process flow diagram of vision-based autonomous robots *source: [44]*

### 2.2.2 Hand-Eye Calibration

Vision-guided robotic systems require an accurate geometric relationship between the camera coordinate system and the robot coordinate frames. Hand–eye calibration [48] is the process of estimating this transformation to enable robots to interpret visual information correctly.

The hand–eye calibration problem aims to determine the rigid transformation between the robot end-effector and the camera coordinate frame. This relationship is typically formulated as the equation AX=XB, where A and B represent the robot and camera motions, respectively, and X is the unknown transformation [49].




**(a)** eye-in-hand configuration **(b)** eye-to-hand configuration

**Figure 2.8:** Hand-Eye calibration configuration *source: [50]*

Hand–eye calibration systems in robotic tasks are generally categorized into two configurations: eye-in-hand [51, 52] and eye-to-hand [53] setups, depending on

whether the camera is mounted on the robot end-effector or fixed in the environment. In both configuration the aim is the identification of the transformation between the Hand, namely the robot base or the end-effector and the Eye which refers to the camera.

**Eye-in-Hand:**

In the eye-in-hand configuration, the vision sensor is mounted directly on the robot end-effector, allowing the camera to move together with the robot during operation. This setup enables the camera to observe objects from different viewpoints as the robot moves through the workspace. As a result, the system can obtain detailed visual information about the target object and its surrounding environment. The eye-in-hand configuration is particularly useful in tasks that require flexible perception and close interaction with objects, such as bin picking, assembly operations, and manipulation of randomly placed parts.

One of the main advantages of the eye-in-hand configuration is the ability to adjust the camera viewpoint dynamically by moving the robot. This allows the system to overcome occlusions and improve object detection accuracy. However, this configuration may also introduce challenges such as motion blur during robot movement and a relatively limited field of view compared to fixed camera setups. In addition, accurate hand–eye calibration becomes essential in this configuration to precisely determine the transformation between the camera and the robot end-effector.

**Eye-to-Hand:**

In the eye-to-hand configuration, the camera is mounted at a fixed position in the robot workspace rather than being attached to the robot itself. The camera observes the robot and the working environment from an external viewpoint, typically from above the workspace. This setup allows the vision system to monitor a larger area and provides a stable image acquisition environment since the camera remains stationary during operation.

The eye-to-hand configuration is widely used in industrial applications such as pick-and-place operations, conveyor tracking, and inspection systems. One of its main advantages is the ability to capture a wide field of view, which allows the system to detect multiple objects simultaneously. Moreover, since the camera does not move, the image acquisition process is generally more stable and less affected by motion-related disturbances. However, this configuration may suffer from occlusions caused by the robot or other objects in the workspace, and it may provide less detailed views of objects compared to the eye-in-hand setup.

### 2.2.3 Robot–Vision Communication Frameworks

**Externally Guided Motion (EGM):**

EGM is a feature provided in ABB industrial robots that enables real-time communication and motion control between the robot controller and an external computer [9].

Through this interface, the robot can exchange feedback data and receive motion commands from an external application, allowing external algorithms to directly influence the robot's motion. Communication in EGM is typically implemented using a UDP-based protocol [54] together with Google Protocol Buffers (Protobuf) [55], enabling low-latency data exchange at update rates of approximately 250 Hz. This high communication frequency allows external systems to continuously update the robot target position based on sensor data or control algorithms.

EGM supports several operational modes that allow different levels of interaction between the external system and the robot controller. The most common modes include *EGM Position Stream*, *EGM Position Guidance*, and *EGM Path Correction*. In *EGM Position Stream*, the robot controller transmits the current and planned positions of the mechanical units to the external device. This mode is primarily used to provide real-time feedback of the robot state to external applications or monitoring systems.

In contrast, *EGM Position Guidance* allows the external system to generate position commands that directly control the robot motion. Instead of following a predefined path programmed in RAPID, the robot continuously updates its position according to the commands received from the external controller. This mode is particularly useful in sensor-based robotic applications where the robot motion must adapt to real-time measurements obtained from external sensors such as cameras or laser scanners. Typical applications include visual servoing, bin picking, and object placement tasks guided by vision systems.

The third mode, *EGM Path Correction*, allows an external system to modify an existing programmed robot path by applying small corrections during execution. In this case, the robot follows the original programmed trajectory while the external device provides incremental adjustments based on sensor measurements. This functionality is commonly used in applications such as seam tracking, where the robot path must be corrected to follow a welding seam or other physical features detected by sensors.

Although EGM provides a flexible interface for integrating external perception and control algorithms with industrial robots, it may bypass certain built-in motion planning and safety features of the robot controller. For this reason, careful system design and extensive testing are required before deploying EGM-based control strategies in real industrial environments. [56]

Figure 2.9 illustrates the general communication architecture of EGM, where an external computer exchanges motion commands and feedback data with the robot controller through a UDP communication interface.

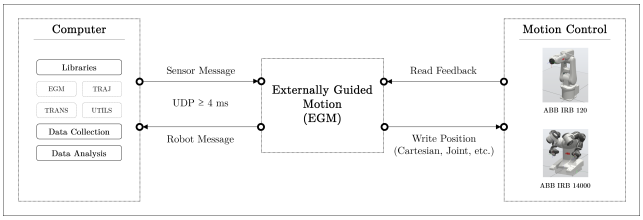



**Figure 2.9:** General communication architecture of EGM, *source: [57]*

### Robot Operating System (ROS):

ROS is an open-source middleware framework widely used for robotic system development and integration [58]. Rather than functioning as a traditional operating system, ROS provides a modular communication infrastructure that enables distributed robotic processes to interact through message-passing mechanisms.

In ROS, computational units are organized as nodes that communicate via topics and services. This architecture supports modularity and rapid system prototyping, making ROS highly popular in academic research and experimental robotics. ROS 2, the second generation of the framework, introduced improvements such as support for the Data Distribution Service (DDS), enhanced scalability, and better suitability for distributed systems [59].

In the context of robot–vision integration, ROS has been extensively used to connect perception pipelines with robotic control modules. Its flexibility and rich ecosystem facilitate rapid integration of sensors, cameras, and motion planning algorithms. However, some issues related to timing, communication delays, and real-time performance have been reported, especially in industrial settings where stable cycle times and synchronized controllers are needed. [60].

As a result, while ROS remains a powerful research-oriented framework, its suitability for tightly constrained industrial production systems requires careful consideration.

# Chapter 3

# Methodology

This chapter describes the methodology used to develop the proposed vision-based robotic manipulation system for industrial object detection and pick-and-place operations. The main objective of the system is to enable reliable object perception and robotic manipulation within a real industrial workspace, where the robot must detect objects and interact with them in a consistent and robust way.

The proposed framework combines computer vision and robotic control within a unified perception-to-manipulation pipeline. The system is organized into two main modules: **Perception** and **Manipulation**.

In the *perception* module, visual data captured by an RGB-D camera is processed in order to detect the target objects and estimate their three-dimensional positions in the workspace. This process allows the system to understand where the objects are located relative to the robot.

The *manipulation* module then uses this spatial information to generate the robot motion commands using the vision-robot communication protocol that are required for the task. Based on this information, the robot is able to execute pick-and-place operations and interact with the objects within the industrial workspace.

The remainder of this chapter is organized as follows. First introduces the overall system architecture and explains how the different components of the system interact with each other. After that, the perception pipeline is described, including the dataset preparation process, the object detection models, and the method used for three-dimensional object localization. This is followed by the manipulation module, which includes the camera-to-robot calibration procedure, motion planning for the robot, and the communication interface between the vision system and the robot controller. Finally, the experimental setup and the evaluation metrics that are used to validate the proposed system are presented.

## 3.1 System Architecture

This section describes the overall architecture of the proposed system, which combines vision-based perception with robotic manipulation for autonomous pick-and-place operations. The structure of the system is shown in Figure 3.1. As illustrated in

the figure, the framework consists of two main stages: the perception stage and the manipulation stage.

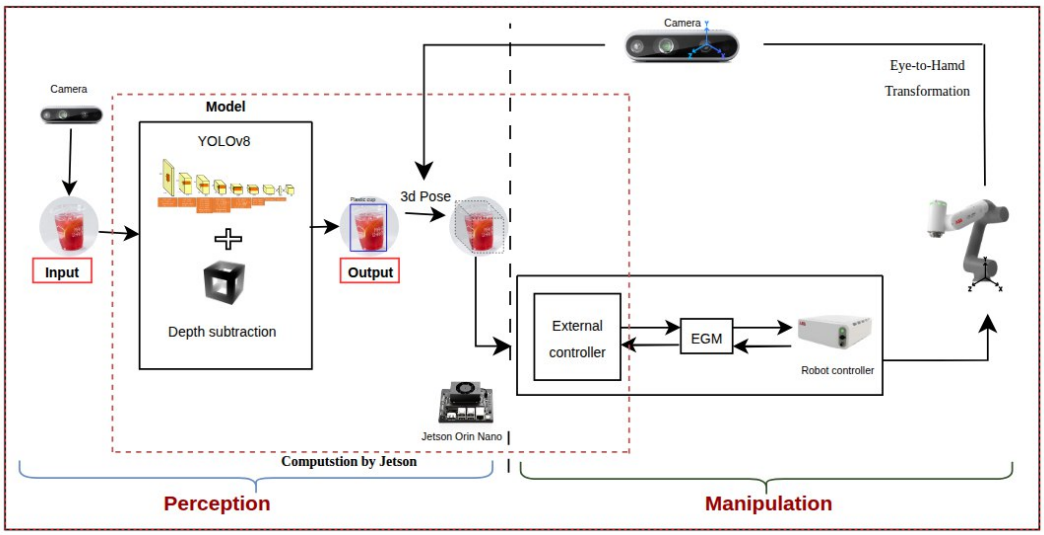



**Figure 3.1:** System architecture of the proposed perception-to-manipulation pipeline.

In the perception stage, an RGB-D camera observes the workspace from a top-down view and captures images of the objects placed on the table. These images are processed using an object detection model based on the YOLOv8 architecture, which identifies the target objects in the scene. To improve detection robustness in the industrial environment, a depth-based subtraction strategy is also applied. This allows the system to detect different types of cups and determine whether a delivery position is occupied.

Along with the 2D detection results produced by the model, depth information is used to estimate the 3D position of each detected object relative to the camera coordinate frame. This spatial information is then used for the subsequent manipulation task.

In the manipulation stage, the estimated object pose is transformed from the camera coordinate frame to the robot base coordinate frame using the calibration parameters obtained through the eye-to-hand calibration procedure. This transformation enables the robot to interpret the object position within its own workspace.

The transformed pose is then sent to an external control application, which generates the corresponding robot motion commands. The external controller communicates with the robot controller through ABB's EGM interface, allowing real-time motion updates based on the detected object position. Using this communication mechanism, the robot performs the pick-and-place task by moving the gripper toward the detected object and executing the grasping and placement actions.

Both the perception module and the external control module pipeline runs on an embedded computing platform (Jetson Orin Nano), where real-time inference is performed and the estimated 3D object position is generated.

The main components of the system are summarized as follows:

- **Industrial Dataset Preparation:** A dataset representing the industrial workspace is collected to train the object detection model.
- **Data Preprocessing and Annotation:** The collected data is processed and annotated to generate labeled samples for training.
- **Model Selection:** An object detection architecture is selected based on detection accuracy and computational efficiency for real-time deployment.
- **Proposed Hybrid Detection Strategy:** A hybrid perception approach combines deep learning-based detection with depth-based processing to improve robustness in industrial environments.
- **3D Localization in Camera Frame:** Depth information is combined with the detection results to estimate the 3D position of the object in the camera coordinate system.
- **Eye-to-Hand Calibration:** A calibration procedure is performed to determine the spatial relationship between the camera and the robot base frame.
- **Cartesian Motion Planning:** The robot trajectory is generated in Cartesian space to move the end-effector toward the detected object.
- **Robot Communication and Control:** The external controller communicates with the robot controller through ABB's EGM interface to execute the planned motion.

Detailed explanations of these components are provided in the following sections.

## 3.2 Perception

### 3.2.1 Industrial Dataset Preparation

Modern object detection models are typically trained on large-scale datasets and demonstrate strong generalization capabilities across many tasks. However, their performance can decrease when applied to specific industrial environments. In such cases, object appearance, lighting conditions, and workspace configurations may differ significantly from the data used during model pretraining. To reduce this domain gap, it is important to collect task-specific data and adapt the detection model to the target environment.

Since no publicly available dataset matched the characteristics of the target industrial scenario, a dedicated dataset was collected directly from the workspace. Figure 3.2 shows sample images from the collected dataset. The dataset focuses on two cup categories used in the application: a *paper cup* and a *plastic cup*.

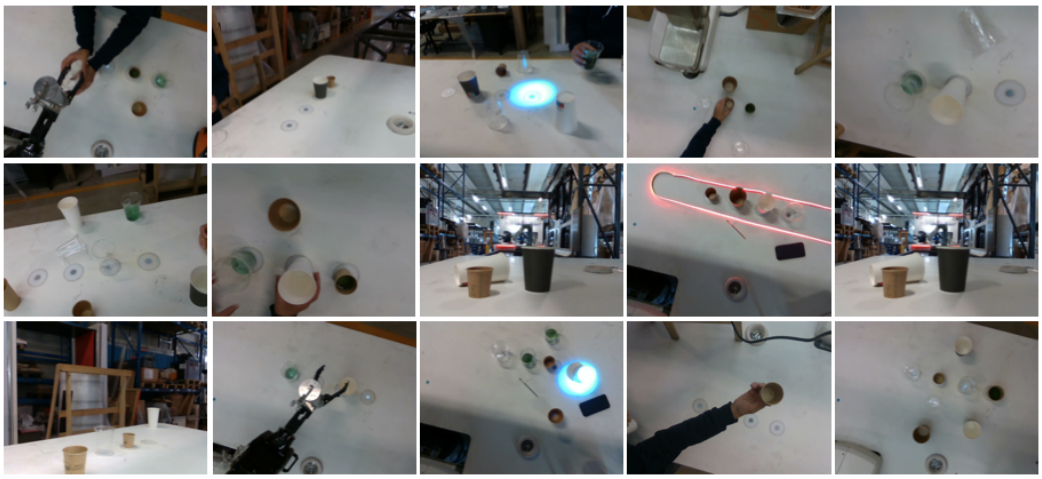


**Figure 3.2:** Sample images from the collected industrial dataset

The data was captured under different workspace configurations, with the objects placed at various positions to introduce natural scene variations. In the final deployment setup, the RGB-D camera is mounted in an eye-to-hand configuration and observes the workspace from a top-down viewpoint. For this reason, most images were captured using a similar overhead perspective to match the expected operating conditions.

To increase robustness against small installation changes, a smaller number of images were also collected with slight camera tilts and viewpoint offsets. The dataset was designed to reflect the real operating conditions of the robotic system. In total, 1,000 RGB images were collected across 5 different acquisition conditions. A summary of the dataset is presented in Table 3.1.

**Table 3.1:** Summary of the collected industrial dataset

| **Item** | **Description** |
|---|---|
| Number of images (RGB) | 1,000 |
| Number of classes | 2 (paper cup, plastic cup) |
| Acquisition conditions | 5 setups (illumination/viewpoint/workspace) |
| Image resolution | *640×480* |

The next subsection describes the preprocessing and annotation steps used to prepare the dataset for training.

### 3.2.2 Data Preprocessing and Annotation

After collecting the raw images from the industrial workspace, several preprocessing steps were performed to prepare the dataset for training the object detection model. These steps included data inspection, image preparation, and annotation of the target objects. The purpose of this stage was to ensure that the dataset was clean, consistent, and suitable for training the detection network.

First, the collected images were visually inspected to verify their quality. Images with severe blur, incorrect exposure, or significant occlusions were removed from the dataset. This step ensures that the training data accurately represents the target objects and reduces the risk of introducing noise into the training process.

Next, the images were prepared for model training. All images were checked to ensure they were sized to a fixed resolution of 640 × 480 pixels to match the input size required by the detection model. This resizing also helps reduce computational cost while keeping enough visual detail for reliable object detection.

The annotation process was performed manually using the Roboflow tool. For each image, bounding boxes were drawn around the target objects and assigned the corresponding class label (*paper cup* or *plastic cup*). The annotations were exported in YOLO format, which stores the normalized coordinates of the bounding boxes together with their class labels.

To speed up the annotation process, the AI-assisted labeling feature in Roboflow was used. The images were uploaded in batches of about 100 samples. First, five images from each batch were manually annotated. Based on these initial annotations, the AI assistant automatically generated annotations for the remaining images in the batch.

Although this approach significantly reduced the manual labeling effort, the automatically generated annotations were not always perfectly accurate. Therefore, all annotations were carefully reviewed and corrected manually to ensure high-quality ground truth labels. High-quality annotations are essential for training reliable object detection models.

Table 3.2 summarizes the main characteristics of the annotation process.

**Table 3.2:** Summary of the dataset annotation process

| **Item** | **Description** |
|---|---|
| Annotation tool | Roboflow |
| Annotation assistance | Roboflow AI assistant |
| Annotation format | YOLO format |

After completing the annotation process, the dataset was divided into different subsets for training and evaluation of the detection model. The details of the dataset split and the training procedure are described in the following sections.

### 3.2.3 Model Selection

Selecting an appropriate object detection model is an important step for achieving reliable perception in robotic manipulation tasks. In industrial environments, the detection model must provide accurate object localization while also maintaining low inference latency to support real-time robot operation. For this reason, several object detection architectures were reviewed and experimentally evaluated before selecting the final model used in this work.

As discussed in the literature review, different detection architectures provide

different trade-offs between detection accuracy and computational efficiency [21]. For robotic systems operating in real time, it is important to find a balance between these two factors.

Traditional two-stage detectors usually achieve high detection accuracy because they separate the region proposal stage from the classification stage. However, this additional processing step increases computational complexity and significantly reduces inference speed. Therefore, these models are generally less suitable for real-time robotic applications where fast perception is required.

In contrast, one-stage detectors perform object localization and classification in a single network pass. Among these approaches, YOLO-based models have shown strong performance in real-time perception tasks due to their efficient architecture. These models can achieve high inference speed while still maintaining competitive detection accuracy, which makes them widely used in robotic perception systems.

Several versions of the YOLO architecture have been introduced in recent years. Although the differences in detection performance among these variants are relatively small, newer versions often provide improvements in model design, training stability, and deployment efficiency. In this work, **YOLOv8** was selected as the base architecture due to its stable performance, active development, and suitability for real-time industrial applications.

Transformer-based detection models and open-vocabulary detectors have also attracted attention in recent years. However, many of these approaches are still relatively new and often require large computational resources. Their higher model complexity usually leads to slower inference speeds, which makes them less suitable for real-time industrial robotic systems at the current stage of development.

Overall, considering the requirements of real-time operation, computational efficiency, and detection reliability, YOLOv8 was selected as the most suitable architecture for the perception module in this work.

### 3.2.4 Proposed Hybrid Perception Strategy

To improve the robustness of the perception system in the industrial workspace, a hybrid perception strategy was introduced. The proposed approach combines several complementary components: a fine-tuned object detector, a zone-based spatial filtering mechanism, depth-based occupancy verification and Temporal Smoothing for Stable Decision. By combining these components, the system can reliably determine whether a delivery spot is occupied with cup or any other objects. At the same time, the approach reduces the effect of noisy detections and small sensor fluctuations that may occur during operation.

#### 3.2.4.1 Fine-Tuning of the Detection Model

To adapt the selected detection model to the visual characteristics of the industrial workspace, the pretrained YOLOv8n model was fine-tuned using the collected dataset. A total of 900 annotated RGB images were used for the training process. The dataset

includes two cup categories captured under different workspace conditions, such as variations in object position, illumination, and minor changes in the surrounding environment.

Since the original YOLOv8 model is trained on large-scale generic datasets, its performance may degrade when applied directly to a specific industrial scenario. Fine-tuning allows the model to adjust its learned feature representations to the visual properties of the target environment. By training the network with images captured directly from the deployment setup, the detector becomes more sensitive to the appearance of the target objects while reducing confusion with background structures present in the workspace.

During the training stage, the pretrained weights were used as the initial network parameters, and the model was further optimized using the collected industrial dataset. This transfer learning strategy significantly reduces the training time while allowing the network to retain the general visual features learned from large-scale datasets. As a result, the fine-tuned model achieves more stable and reliable detection performance under the operating conditions of the robotic system.

To monitor the training process and reduce the risk of overfitting, the dataset was divided into training and validation subsets. The validation set was used to evaluate the model performance during training and ensure tha tthe detector generalizes well to unseen images from the workspace.

**Zone-based detection strategy:** To further improve the robustness of the perception system and reduce spatial noise in the detection results, a zone-based detection strategy was introduced. Instead of directly relying on raw object detections in the image, the workspace was divided into a set of predefined delivery zones. Each zone corresponds to a specific delivery spot in the physical workspace where an object may be placed and where the robot is expected to perform a manipulation task.

In the proposed system, each zone $Z_i$ is represented as a circular region defined by its center coordinates $(x_i, y_i)$ and a radius $r_i$ in the image plane. These parameters are manually configured based on the geometry of the workspace and the locations of the delivery spots.

During runtime, the YOLO detector processes the RGB frame and generates bounding boxes for detected objects. For each detection, the center point of the bounding box $(c_x, c_y)$ is computed. The system then determines whether the detected object belongs to a specific delivery zone by checking if the center point lies within the circular region $Z_i$.

Mathematically, a detection is assigned to zone $Z_i$ if the following condition is satisfied:

$$(c_x - x_i)^2 + (c_y - y_i)^2 \leq r_i^2 \tag{3.1}$$

where $(c_x, c_y)$ represents the center of the detected bounding box, $(x_i, y_i)$ denotes the center of the zone, and $r_i$ is the radius of the zone.

If this condition holds, the corresponding zone is considered to contain the detected

object. This spatial filtering mechanism ensures that only detections occurring within the predefined delivery areas are considered valid.

The use of zone-based detection provides two main advantages. First, it reduces the influence of noisy detections outside the regions of interest. Second, it allows the perception system to convert object detections into discrete zone-level signals, which can be directly used by the robot controller to determine whether a particular delivery spot is occupied.

#### 3.2.4.2 Depth-Based Differences Approach

One of the requirements defined at the beginning of the project is that the perception module should not only detect the cup, but also determine whether a given delivery spot is *occupied*. Since the RGB-D camera provides depth measurements, an occupancy verification mechanism based on *height change* (depth variation) was implemented.

For each delivery zone $Z_i$, a reference depth frame $D_{\text{base}}(x, y)$ is captured once while the workspace is empty. During operation, each incoming depth frame $D_{\text{cur}}(x, y)$ is compared against this baseline to identify meaningful geometric changes inside the zone. Invalid depth samples (zero values) are discarded before computing statistics.

For each zone, the mean depth is computed for both the baseline and the current frame as

$$\bar{D}^i_{\text{base,cur}} = \frac{1}{N^i_{\text{base,cur}}} \sum_{(x,y)\in Z_i} D_{\text{base,cur}}(x, y), \tag{3.2}$$

where $N^i$ denotes the number of valid depth pixels in $Z_i$. The absolute depth difference is then calculated as

$$\Delta_i = \left| \bar{D}^i_{\text{cur}} - \bar{D}^i_{\text{base}} \right|. \tag{3.3}$$

A delivery zone is considered *occupied* when the depth change exceeds a predefined threshold:

$$\Delta_i > \tau_d, \tag{3.4}$$

where $\tau_d$ is set to 10 mm in the current implementation. This threshold acts as a simple but effective criterion to distinguish meaningful object presence from minor sensor noise.

**Temporal smoothing for stable decisions:** Depth measurements may fluctuate across frames due to sensor noise, reflections, or temporary occlusions. To ensure stable occupancy decisions, a short temporal filter was applied using a sliding window of recent frames. For each zone, a binary occupancy signal is stored over a buffer of length $B = 5$. A zone is activated only if at least 60% of the buffered frames indicate occupancy (i.e., at least 3 out of 5 frames). This majority-voting mechanism reduces flickering while preserving low-latency responsiveness required in real-time robotic

operation.

In summary, the presence of an object in a delivery zone is determined by combining YOLO-based object detection with depth-based height change analysis. The YOLO detector is responsible for identifying the target object in the RGB image and estimating its location within the predefined delivery zones. However, visual detections alone may occasionally lead to false positives due to lighting variations or background structures. To address this limitation, the depth information provided by the RGB-D camera is used to verify the physical presence of an object by measuring the height variation relative to a reference depth frame of the empty workspace.

By integrating these two complementary sources of information, the system is able to reliably determine whether a delivery spot is occupied. This hybrid perception strategy improves robustness against visual noise while maintaining the real-time performance required for robotic manipulation tasks.

### 3.2.5 3D Localization in Camera Frame

After detecting the cup in the RGB image, its three-dimensional position is estimated in the camera coordinate frame using the aligned depth stream of the RGB-D sensor. For each detection, the center of the predicted bounding box is computed as

$$u = \frac{x_1 + x_2}{2}, \qquad v = \frac{y_1 + y_2}{2}, \tag{3.5}$$

where $(x_1, y_1)$ and $(x_2, y_2)$ denote the top-left and bottom-right corners of the bounding box in pixel coordinates.

Given the pixel location $(u, v)$, the depth value $d$ is obtained from the depth frame at the same location. Since the depth stream is aligned to the RGB image, the retrieved depth corresponds to the same point in the scene. Invalid depth measurements (e.g., zero values) are discarded.

To convert the 2D pixel location into a 3D point, the camera intrinsic parameters $(f_x, f_y, c_x, c_y)$ are used to deproject the pixel into the camera frame. The resulting 3D coordinates $(X_c, Y_c, Z_c)$ are computed as

$$X_c = \frac{(u - c_x)\, d}{f_x}, \qquad Y_c = \frac{(v - c_y)\, d}{f_y}, \qquad Z_c = d, \tag{3.6}$$

where $d$ is the depth value expressed in meters. In practice, the camera driver provides the intrinsic parameters required for this deprojection process. Using these parameters, the pixel location together with its corresponding depth value can be directly converted into a 3D point in the camera coordinate frame.

The output of this step is the 3D position of the detected cup expressed in the camera coordinate system. This position is later transformed into the robot base frame through the eye-to-hand calibration described in the next section.

## 3.3 Manipulation

### 3.3.1 Eye-to-Hand Calibration

**Eye-to-Hand Calibration:**

During manipulation, the perception system detects the cup and estimates its 3D position in the camera coordinate frame. The procedure used to compute this 3D point from RGB and depth data is described in Section 3.2.5.

To enable robotic manipulation, the estimated object position must be transformed from the camera coordinate frame to the robot base frame. This transformation is defined by the Eye-to-Hand calibration matrix.

Given the homogeneous representation of the detected point

$$P_{\text{camera}} = \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{3.7}$$

Given the fixed Eye-to-Hand configuration, the transformation to the robot base frame is:

$$P_{\text{robot}} = T_{rc} P_{\text{camera}}, \quad T_{rc} = \begin{bmatrix} R_{rc} & t_{rc} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.8}$$

**Geometry-Based Calibration:** Geometry-Based Calibration: Since the camera is mounted rigidly and perpendicular to the tabletop, the camera–robot transformation can be derived analytically from the known mechanical configuration. The camera $x$-axis aligns with the robot $x$-axis, while the $y$ and $z$ axes are inverted due to the downward-facing orientation, and the translation vector, determined from the mechanical layout and CAD measurements, resulting in:

$$R_{rc} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, t_{rc} = \begin{bmatrix} -93 \\ -752 \\ 900 \end{bmatrix} \text{ mm} \tag{3.9}$$

The translation vector, determined from the mechanical layout and CAD measurements, is:

Thus, the complete homogeneous transformation is:

$$T_{\text{camera}\to\text{robot}} = \begin{bmatrix} 1 & 0 & 0 & -93 \\ 0 & -1 & 0 & -752 \\ 0 & 0 & -1 & 900 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

This geometry-based calibration leverages the known mounting configuration to reduce complexity while maintaining high accuracy, eliminating the need for iterative custom eye-to-hand optimization.

### 3.3.2 Cartesian Motion Planning

Once the target object position is estimated in the robot base frame, the next step is to generate a motion command for the robot end-effector. Instead of relying on predefined trajectories, a step-based Cartesian motion strategy is used to incrementally move the robot toward the target position.

Let $p_k \in \mathbb{R}^3$ denote the current end-effector position at control cycle $k$, and let $p^*$ represent the desired target position obtained from the perception module. The Cartesian position error is computed as

$$e_k = p^* - p_k. \tag{3.11}$$

To ensure stable motion and avoid sudden jumps in the commanded position, the motion increment is bounded using a maximum velocity constraint. The admissible motion increment is defined as

$$\Delta_k = \min\left(\|e_k\|,\ v_{\max}\, dt\right), \tag{3.12}$$

where $v_{\max}$ denotes the maximum allowed Cartesian velocity and $dt$ is the control cycle time.

The next commanded position is then computed as

$$p_{k+1} = p_k + \Delta_k \frac{e_k}{\|e_k\|}. \tag{3.13}$$

This formulation ensures that the end-effector moves smoothly toward the target while respecting the predefined velocity limits. During the motion execution, the end-effector orientation is kept constant, and only the Cartesian position is updated. The motion is considered complete when the position error satisfies

$$\|e_k\| < \epsilon, \tag{3.14}$$

where $\epsilon$ is a predefined tolerance threshold.

### 3.3.3 Robot Communication and Control

Several approaches can be used to establish communication between a perception system and an industrial robot. A common solution in robotics research is to use the ROS, which provides a flexible framework for integrating perception, planning, and control modules.

For this reason, an initial investigation was conducted to evaluate the feasibility of implementing the system using ROS. However, at the time of development, an official MoveIt driver for the ABB GoFa robot was not available. This limitation made it difficult to reliably integrate the robot with the ROS-based control framework.

As a result, an alternative communication approach was adopted. The *EGM* interface, officially provided by ABB, was selected as the primary communication

method. Unlike ROS-based solutions, which are commonly used in research environments, EGM is an industrial communication interface designed specifically for real-time robot control. This makes it a more reliable and robust solution for industrial applications.

**EGM implementation:** The robotic manipulation system consists of an RGB-D camera, an external control computer, and an ABB industrial robot equipped with the EGM communication interface. The RGB-D camera is used to capture both color and depth images of the workspace, which are processed by the perception module to detect the target object and estimate its three-dimensional position.

The perception and control algorithms run on an external computer connected to the robot controller through a local Ethernet network. Communication between the external computer and the robot controller is established using the EGM interface provided by ABB.

Before running the system, the EGM functionality was enabled on the robot controller following the configuration procedure described in the ABB documentation. In this setup, the robot controller was configured to listen to a UDP unicast device, allowing an external computer to send motion commands directly to the robot through network messages.

After enabling the EGM interface, the host computer was registered as a UDP unicast device. This configuration allows bidirectional communication between the robot controller and the external control program. During operation, the robot continuously sends feedback messages containing its current joint positions and TCP pose, while the external controller sends updated motion commands.

To implement this communication in the control software, the open-source Python library `ABB-EGM-Python` was used [61]. This library provides a convenient interface for receiving robot state messages and sending motion updates through the EGM protocol.

In the proposed system, the perception module first detects the cup and estimates its 3D position in the robot base coordinate frame. This position is then passed to the external control module, which computes incremental Cartesian motion commands for the robot end-effector which described in Section 3.3.2.

The control program continuously runs a communication loop where it receives the current robot state, computes a new target pose, and transmits the updated Cartesian setpoint to the robot controller through the EGM interface. This high-frequency communication loop enables smooth and responsive robot motion, allowing the robot to gradually converge toward the detected object during the pick-and-place operation.

## 3.4 Experimental Setup

This section describes the experimental setup used to evaluate the proposed vision-guided robotic manipulation system. The setup includes the hardware components, software environment, and workspace configuration used during the experiments.

### 3.4.1 Hardware System

The experimental system consists of three main hardware components: an RGB-D camera for visual perception, an embedded computing platform for real-time processing, and an industrial robotic manipulator for executing the pick-and-place tasks.

**RGB-D Camera:**

An Intel RealSense D435 RGB-D camera is used to capture both color and depth information of the workspace. RGB-D sensors provide synchronized RGB images together with per-pixel depth measurements, which makes them particularly suitable for robotic perception tasks that require spatial information.

**Figure 3.3:** Intel RealSense D435 RGB-D camera

In the proposed system, the camera is mounted above the workspace in an eye-to-hand configuration, providing a top-down view of the delivery area. This configuration allows the perception module to observe the entire workspace and detect objects placed on the table. The depth stream provided by the RealSense sensor is also used to estimate the three-dimensional position of the detected objects and to verify the occupancy of the delivery zones using depth difference analysis.

The Intel RealSense camera was selected due to its reliable depth sensing, compact form factor, and straightforward integration with the RealSense SDK, which provides direct access to aligned RGB and depth streams.

**Embedded Computing Platform:**

All perception and control algorithms are executed on an NVIDIA Jetson Orin Nano embedded computer. The Jetson platform is designed for edge AI applications and provides GPU acceleration for deep learning inference.

In this project, the Jetson device runs the object detection model, processes the RGB and depth data streams from the camera, and performs the hybrid detection strategy described in the previous sections. The embedded platform was selected due to its ability to perform real-time vision processing while maintaining a compact and energy-efficient hardware suitable for robotic systems.

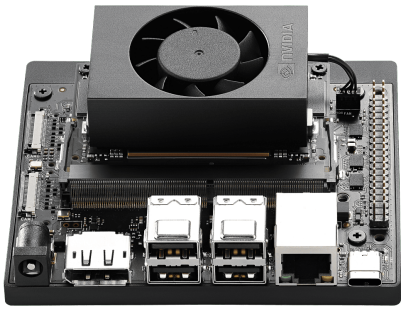


**Figure 3.4:** NVIDIA Jetson Orin Nano

**Robotic Manipulator:**

The manipulation tasks are performed using an ABB GoFa collaborative robotic arm equipped with a parallel gripper.

The ABB GoFa robot is frequently used in industrial robotic tasks due to its high positioning accuracy, flexible control interface, and compatibility with ABB's EGM protocol. This interface allows external systems to send Cartesian motion updates to the robot controller with low latency, enabling the integration of vision-based feedback for real-time manipulation tasks.

**Figure 3.5:** ABB GoFa

### 3.4.2 Software Environment

The software components of the proposed system were implemented on the NVIDIA Jetson Orin Nano embedded platform using Jetson Linux 36.4, which is part of the JetPack 6.1 software stack. This environment is based on Ubuntu 22.04 and provides integrated GPU acceleration through CUDA, enabling efficient execution of deep learning inference and real-time image processing tasks.

**Table 3.3:** Software components used in the experimental setup

| Component | Technology |
|---|---|
| Operating System | Jetson Linux 36.4 (JetPack 6.1) |
| Programming Language | Python 3.10 |
| Object Detection | Ultralytics YOLOv8 |
| Camera Driver | Intel RealSense SDK (pyrealsense2) |
| Robot Programming | RAPID |

The perception pipeline was developed in Python 3.10. Object detection was implemented using the Ultralytics YOLOv8 framework, which provides an efficient implementation of the YOLO architecture suitable for real-time applications on embedded platforms.

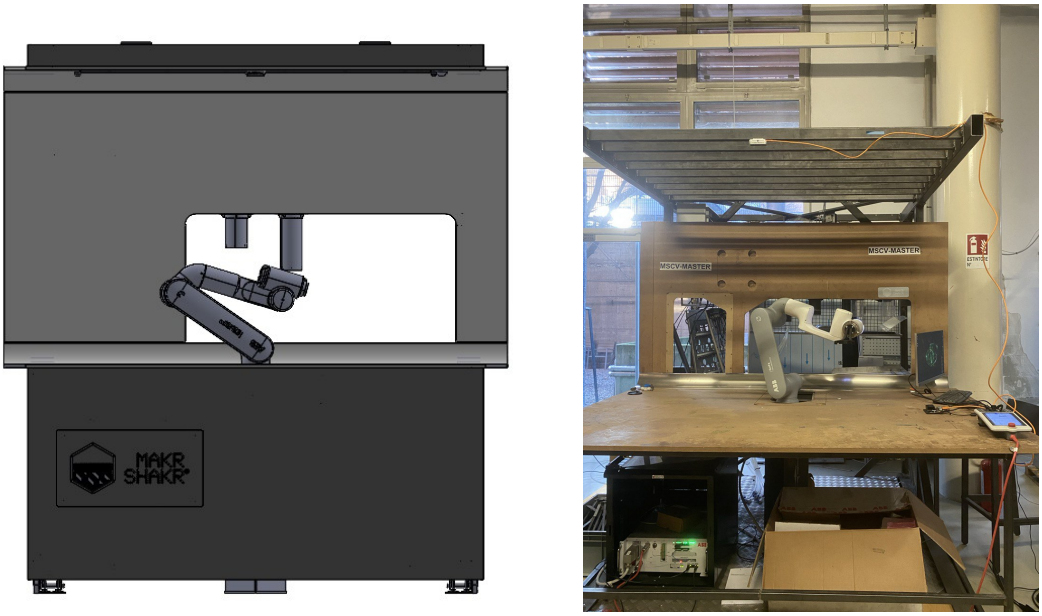
Depth and RGB acquisition and processing were performed using the Intel RealSense SDK through the `pyrealsense2` Python interface. This library provides direct access to synchronized RGB and depth streams from the RealSense camera and allows pixel-wise depth retrieval for 3D localization of detected objects.

Finally, The Communication with the ABB GoFa robot was implemented using the ABB robot controller environment. The robot programs were developed using the RAPID programming language and tested in RobotStudio, which allows robot motions deployment on the physical system.

Table 3.3 summarizes the main software components used in the experimental system.

### 3.4.3 Workspace Configuration

The experiments were conducted in a table-top workspace designed for vision-guided pick-and-place operations. The workspace consists of a flat surface where the target objects are placed and manipulated by the robotic arm.




**(a)** Simulation version **(b)** Real-world version

**Figure 3.6:** Workspace configuration

The RGB-D camera is mounted above the workspace in an eye-to-hand configuration, providing a top-down view of the entire working area. This camera placement allows the perception module to observe all delivery zones simultaneously while maintaining consistent visibility of the objects on the table. The aligned RGB and depth streams obtained from the camera are used for object detection, depth verification, and 3D localization.

Within the workspace, several predefined delivery zones are defined to represent potential object placement locations. Each zone corresponds to a circular region in the image plane and is associated with a specific delivery spot on the table. These zones are used by the perception system to determine whether an object is present in a given location.

The ABB GoFa robot is positioned adjacent to the workspace such that its reachable manipulation area fully covers the defined delivery zones. This configuration enables the robot to perform pick-and-place operations based on the target positions provided by the perception module.

### 3.4.4 Evaluation Metrics

To evaluate the performance of the proposed system, two levels of evaluation are considered: vision evaluation, and manipulation evaluation.

**Vision evaluation**:

The object detection performance is measured using standard computer vision metrics. In the following section, each evaluation metric is explained in detail.

***Precision*** measures how many of the detected objects are correct detections. It is defined as:

$$Precision = \frac{TP}{TP + FP}$$

where $TP$ represents true positive detections and $FP$ represents false positives.

***Recall*** measures how many of the actual objects present in the scene are successfully detected by the model:

$$Recall = \frac{TP}{TP + FN}$$

where $FN$ represents false negatives.

***Mean Average Precision (mAP)*** summarizes the overall detection performance by evaluating the precision–recall relationship of the detector. It reflects both the classification accuracy and the localization quality of the predicted bounding boxes. ***mAP@0.5*** computes the mean average precision using an Intersection over Union (IoU) threshold of 0.5, meaning that a detection is considered correct if the overlap between the predicted bounding box and the ground truth box is at least 50%. ***mAP@0.5:0.95*** is a stricter metric that averages the detection performance over multiple IoU thresholds ranging from 0.5 to 0.95, providing a more comprehensive evaluation of localization accuracy.

**Computational Performance** In addition to detection accuracy, the computational performance of the models is also evaluated to assess their suitability for real-time robotic applications.

**Number of Parameters.** The number of parameters indicates the total amount of learnable weights in the neural network. Models with fewer parameters generally require less memory and may offer faster inference, which is important for deployment on embedded platforms.

**GFLOPs.** Giga Floating Point Operations (GFLOPs)represent the number of floating-point operations required to process a single input image. This metric provides an estimate of the computational complexity of the model.

**Inference Time.** Inference time measures the time required for the model to process an input image and produce the detection results. This metric is particularly important for real-time robotic systems, where fast response is required during task execution.

### Manipulation evaluation:

The overall robotic manipulation performance is evaluated using the pick-and-place success rate:

$$Success\ Rate = \frac{\text{Successful pick-and-place operations}}{\text{Total trials}}$$

A successful trial means that the robot correctly detects the object, reaches the grasping position, and completes the pick-and-place operation without failure.

# Chapter 4

# Results and Discussion

In this chapter, the experimental results obtained from the implementation of the proposed vision-guided robotic manipulation system are presented and analyzed. The purpose of these experiments is to evaluate the performance of the perception module and to examine how well the complete robotic system works in practical scenarios.

For clarity, the results are divided into two main parts. First, the performance of the perception module is evaluated using the collected dataset. This part includes the training configuration of the object detection model, the quantitative detection results, the runtime performance, and the evaluation of the proposed hybrid detection strategy.

Second, the manipulation performance of the robotic system is evaluated through real-world pick-and-place experiments using the ABB GoFa robotic arm. In these experiments, the system is tested on its ability to detect the cups, estimate their 3D positions, and perform reliable pick-and-place operations within the workspace.

Finally, the results are discussed to highlight the strengths and limitations of the proposed approach and to better understand the overall behavior of the integrated perception and manipulation pipeline.

## 4.1 Perception Task

This section presents the experimental results obtained from training and evaluating three YOLOv8 model variants on the custom cup detection dataset. The goal of this evaluation is to analyze the detection performance of each model and determine the most suitable architecture for the perception module of the robotic system.

The models are evaluated using standard object detection metrics, including Precision, Recall, mAP@0.5, and mAP@0.5:0.95. In addition, the number of parameters, GFLOPs, and inference time are reported to evaluate the computational complexity and runtime performance of each model, allowing a comparison between detection accuracy and real-time suitability for robotic applications.

### 4.1.1 Training Setup and hyperparameters

The experiments were carried out using the Ultralytics YOLOv8 framework. Three versions of the model were fine-tuned on the custom dataset: YOLOv8n (nano), YOLOv8s (small), and YOLOv8l (large). These models mainly differ in their network depth and number of parameters, which influences both their detection accuracy and computational cost.

The dataset contains images of two object classes: *paper cup* and *plastic cup*. The images were collected directly from the robotic workspace and include variations in object position, orientation, and lighting conditions in order to represent realistic operating scenarios.

To ensure a fair comparison between the evaluated models, the same training configuration was used in all experiments. The main hyperparameters were selected through a search process in which different parameter combinations were tested during training. In this process, a predefined search space was defined for several key training parameters, and multiple configurations were evaluated to identify the settings that resulted in stable convergence and strong validation performance.

The final hyperparameter configuration used in this study is summarized in Table 4.1. These parameters control different aspects of the training process.

**Table 4.1:** Training hyperparameters used for YOLOv8 model training

| **Hyperparameter** | **Value** |
|---|---|
| Epochs | 120 |
| Batch Size | 16 |
| Image Size | 640 × 480 |
| Initial Learning Rate | 0.01 |
| Momentum | 0.937 |
| Weight Decay | 0.0005 |
| Optimizer | AdamW |

The number of training epochs determines how many times the entire training dataset is passed through the network during training. A sufficient number of epochs allows the model to learn meaningful visual features from the data, while too many epochs may increase the risk of overfitting.

The batch size specifies how many training samples are processed in a single iteration. This parameter affects both the stability of gradient updates and the computational efficiency of the training process.

The learning rate controls how large the parameter updates are during optimization. Selecting an appropriate learning rate is important to achieve stable convergence while avoiding unstable training behavior.

Momentum is used to accelerate gradient descent by incorporating information from previous updates. This helps the optimization process move more consistently toward an optimal solution.

Weight decay acts as a regularization technique that penalizes excessively large model weights. This helps improve the generalization ability of the model and reduces

the risk of overfitting.

In addition to these parameters, data augmentation techniques such as mosaic augmentation and horizontal flipping were applied during training. These techniques increase the diversity of the training data and improve the robustness of the object detection model.

### 4.1.2 Quantitative Results

Table 4.2 presents the detection performance of the three YOLOv8 models on the validation dataset.

As shown in Table 4.2, the YOLOv8s model achieved the best overall detection performance. It obtained the highest Precision (0.992), Recall (0.982), and mAP values among the evaluated models.

Interestingly, the larger YOLOv8l model, despite having significantly more parameters, did not outperform YOLOv8s on this dataset. This suggests that the medium-sized YOLOv8s model provides a better balance between model complexity and detection accuracy for this specific task.

**Table 4.2:** Detection performance comparison of YOLOv8 models

| Model | Precision | Recall | mAP@0.5 | mAP@0.5:0.95 |
|---|---|---|---|---|
| YOLOv8n | 0.980 | 0.975 | 0.990 | 0.919 |
| **YOLOv8s** | **0.992** | **0.982** | **0.994** | **0.941** |
| YOLOv8l | 0.986 | 0.976 | 0.988 | 0.928 |

### 4.1.3 Computational Efficiency and Runtime Analysis

In addition to detection accuracy, it is also important to evaluate the computational complexity of each model, especially for real-time robotic applications. Table 4.3 presents a comparison of the three YOLOv8 variants in terms of the number of parameters, GFLOPs, and inference time.

The number of parameters indicates the size of the neural network, while GFLOPs represent the computational complexity required for a single forward pass. Inference time measures how long the model takes to process one image, which is a key factor for real-time deployment.

As shown in Table 4.3, YOLOv8n is the smallest model with only 3.0 million parameters and the lowest computational complexity (8.1 GFLOPs). As a result, it achieves the fastest inference time of 2.2 ms per image, making it suitable for systems with limited computational resources.

YOLOv8s provides a balance between computational cost and detection performance. Although it contains more parameters than YOLOv8n, its inference time remains relatively low while achieving the best detection accuracy among the evaluated models.

In contrast, YOLOv8l has a significantly larger network size with 43.6 million parameters and a much higher computational complexity (164.8 GFLOPs). This

leads to a longer inference time of 16.8 ms per image. However, despite the increased model size and computational cost, YOLOv8l did not show a noticeable improvement in detection performance compared to YOLOv8s.

Explanation for this behavior is the relatively limited dataset size used in this study (1000 images). Larger models typically require more training data to fully utilize their representational capacity. Therefore, the medium-sized YOLOv8s model achieved the best trade-off between accuracy and computational efficiency for this task.

**Table 4.3:** Complexity comparison of YOLOv8 model variants

| Model | Parameters | GFLOPs | Inference Time (ms) |
|---|---|---|---|
| YOLOv8n | 3.0M | 8.1 | 2.2 |
| YOLOv8s | 11.1M | 28.4 | 5.4 |
| YOLOv8l | 43.6M | 164.8 | 16.8 |

### 4.1.4 Test Set Evaluation

To further evaluate the generalization capability of the trained model, the YOLOv8n model was tested on a separate test dataset consisting of 100 images.

The evaluation results show that the model achieves a precision of 0.990 and a recall of 0.979. The mean Average Precision reaches 0.994 at an IoU threshold of 0.5 (mAP@0.5), while the more strict metric mAP@0.5:0.95 reaches 0.914.

These results indicate that the model is able to accurately detect cup objects with both high precision and high recall on previously unseen data. The high mAP values also suggest that the predicted bounding boxes closely match the ground truth annotations.

When evaluated per class, the model achieves mAP@0.5:0.95 scores of 0.919 for the *paper cup* class and 0.908 for the *plastic cup* class, demonstrating consistent detection performance across both object categories.

**Table 4.4:** Per-class detection performance on the test dataset

| Model | Class | Precision | Recall | mAP@0.5 | mAP@0.5:0.95 |
|---|---|---|---|---|---|
| YOLOv8n | paper cup | 0.982 | 0.980 | 0.992 | 0.920 |
| | plastic cup | 0.999 | 0.978 | 0.995 | 0.908 |
| YOLOv8s | paper cup | 0.982 | 0.981 | 0.992 | **0.928** |
| | plastic cup | 0.995 | 0.980 | 0.995 | **0.915** |
| YOLOv8l | paper cup | 0.970 | 0.982 | 0.991 | 0.925 |
| | plastic cup | 0.995 | 0.994 | 0.995 | 0.908 |

Table 4.4 presents the per-class detection performance of the evaluated YOLOv8 models on the test dataset.

The results show that all models achieve high precision and recall values for both object categories, indicating reliable detection performance. Among the evaluated models, YOLOv8s achieves the highest mAP@0.5:0.95 for both the *paper cup* and *plastic cup* classes, reaching 0.928 and 0.915 respectively.

The results also demonstrate that the detection performance is consistent across the two classes, with only minor differences between them. This indicates that the trained models are able to effectively distinguish between the two cup categories without significant class imbalance.

Although YOLOv8l is the largest model in terms of network capacity, it does not significantly outperform YOLOv8s. This suggests that the medium-sized YOLOv8s model provides the best trade-off between detection accuracy and computational efficiency for this dataset.

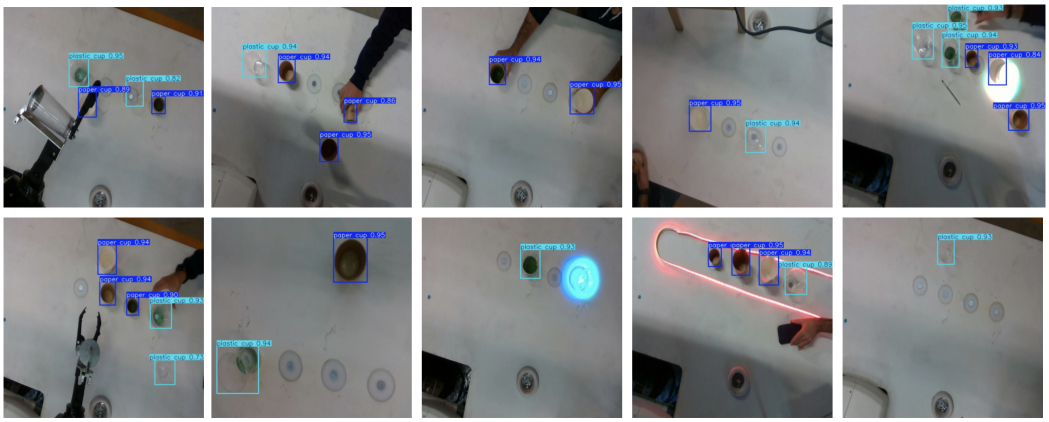


**Figure 4.1:** Sample images from the detection of test set using YOLOv8s

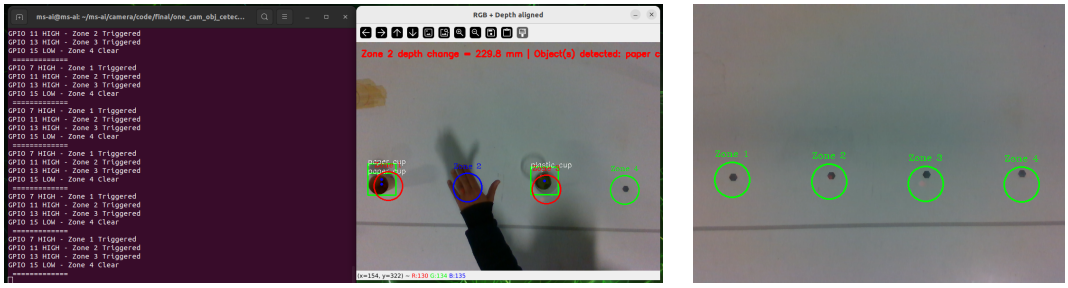
### 4.1.5 Hybrid Detection Evaluation

In addition to the object detection model, the proposed system employs a hybrid perception strategy that incorporates depth-based subtraction analysis. This approach aims to improve the robustness of detecting cups and determining whether a delivery zone is occupied by any object.

To evaluate the effectiveness of the proposed method, experiments were conducted by comparing the YOLO-based object detection alone with the proposed hybrid approach that incorporates depth verification.

The hybrid detection strategy was evaluated using four different cup types placed in various delivery zones: a small paper cup, a large paper cup, a transparent plastic cup, and a transparent martini cup. In addition, two other objects, a human hand and a mobile phone, were included in the experiments, as these objects frequently appear in the human–robot collaborative workspace. This evaluation helps assess the system's ability to determine whether a delivery spot is occupied by any object.

Figure 4.3 presents examples of cup detection results for different cup types. Figure 4.2 illustrates the final hybrid detection results across the delivery zones.



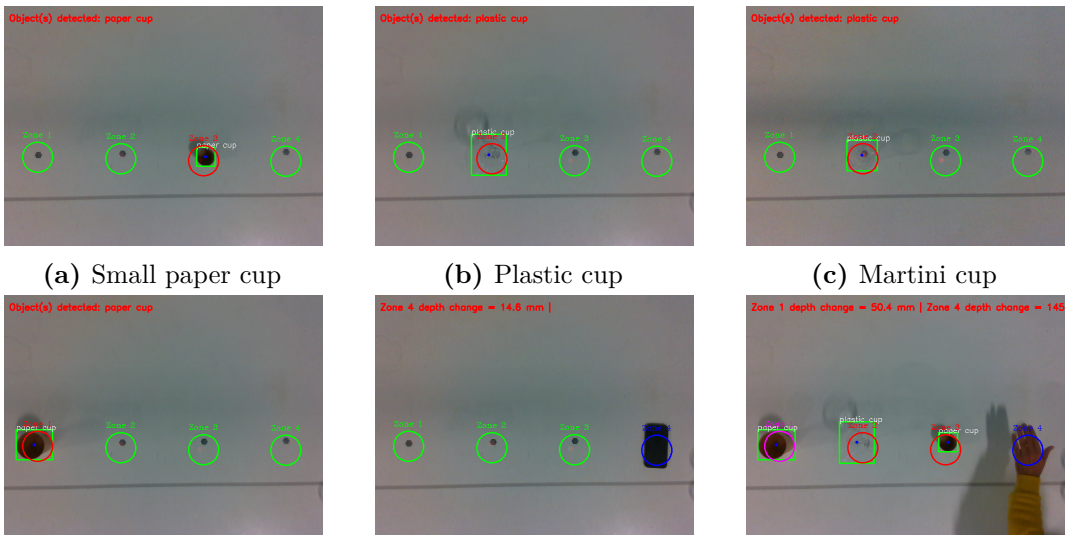
**(a)** Final result **(b)** Base form

**Figure 4.2:** Hybrid detection evaluation

The results show that the detection model achieves an accuracy consistent with the results reported in the previous section. When the hybrid strategy is applied, the system can also detect object presence through depth-based height variation even when the cup is not detected by the visual model.

Specifically, if the height difference between the current depth measurement and the reference empty workspace exceeds 10 mm, the system classifies the delivery zone as occupied. As illustrated in Figure 4.3(e), height changes greater than 30 mm are detected with nearly 100% reliability, while height variations between the threshold (10 mm) and 30 mm may occasionally lead to detection errors of up to 20%.

In Figure 4.3(f), three colors are used to visualize the detection outcomes. Red indicates successful cup detection by the object detection model. Blue represents occupancy detection based solely on the depth-based height difference. Purple indicates cases where both detection mechanisms identify the object simultaneously, as observed for Zone 1.



**(a)** Small paper cup **(b)** Plastic cup **(c)** Martini cup

**(d)** Big paper cup **(e)** Depth detection **(f)** All in one

**Figure 4.3:** Different scenario for hybrid approach

## 4.2 Manipulation Performance

To better evaluate the robustness of the proposed perception and manipulation system, additional experiments were conducted in different regions of the workspace. In real robotic environments, the accuracy of visual perception can be affected by several factors, such as camera distortion, perspective changes, and partial visibility of objects. These effects usually become stronger near the edges of the camera's field of view.

To analyze this effect, the tabletop workspace was divided into three zones: *left*, *middle*, and *right*. This division allows the system performance to be evaluated in different spatial regions relative to the camera viewpoint. Objects placed near the sides of the image are more likely to be affected by distortion or partial visibility compared to objects located in the center.

For the experiments, three different types of cups were used: *paper cup*, *plastic cup*, and *martini cup*, as shown in Figure 4.4. These objects were selected because they have different shapes and visual properties and are commonly used in the company's production line. Such variations may influence both object detection and grasping performance.


**(a)** Paper cup **(b)** Plastic cup **(c)** Martini cup

**Figure 4.4:** Different cup types used in the pick-and-place experiments

For each zone, pick-and-place experiments were performed using all three cup types. Each configuration was tested for 15 trials in order to evaluate the stability and reliability of the perception and manipulation system.

**Table 4.5:** Pick-and-place performance across different workspace zones

| Zone | Object Type | Trials | Success | Failures | Success Rate |
|---|---|---|---|---|---|
| Left Side | Paper Cup | 15 | 14 | 1 | 93.33 |
| | Plastic Cup | 15 | 12 | 3 | 80 |
| | Martini Cup | 15 | 15 | 0 | 100 |
| Middle | Paper Cup | 15 | 15 | 0 | 100 |
| | Plastic Cup | 15 | 14 | 1 | 93.33 |
| | Martini Cup | 15 | 15 | 0 | 100 |
| Right Side | Paper Cup | 15 | 13 | 2 | 93.33 |
| | Plastic Cup | 15 | 12 | 3 | 80 |
| | Martini Cup | 15 | 15 | 0 | 100 |

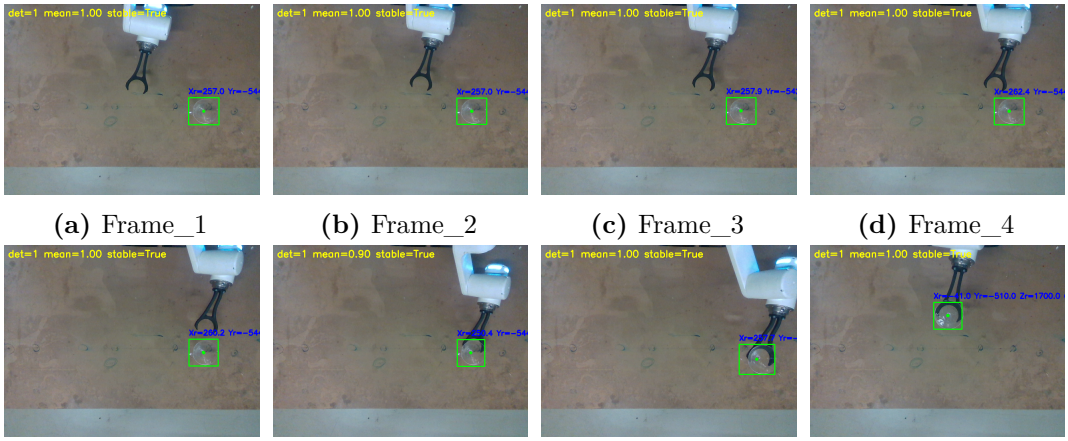
The results of the pick-and-place experiments across different workspace zones are presented in Table 4.5. The experiments were conducted across three object types and three workspace regions, with 15 trials per configuration.

Overall, the system demonstrates stable and reliable performance across the tested conditions. As expected, the middle region of the workspace shows the most consistent results. Objects placed in this region are closer to the center of the camera's field of view and, therefore, less affected by perspective distortion or partial visibility. As a result, the success rate in the middle zone reaches 100% for the paper cup and martini cup, and 93.33% for the plastic cup.

In contrast, slightly lower success rates are observed in the left and right regions of the workspace. These areas are located near the edges of the camera view, where camera distortion and partial visibility of the objects can influence the perception system and the accuracy of the estimated grasping position.

Among the tested object types, the martini cup achieved a 100% success rate in all zones. This behavior can be explained by the geometric structure of the martini cup, which includes a stable base that allows the robot gripper to successfully grasp the object even when small perception or positioning errors occur.

The paper cup also shows high success rates across all zones, with only a small number of failures. In contrast, the plastic cup exhibits slightly lower success rates, particularly in the side regions of the workspace. Two main factors contribute to these failures. First, in some cases the object detection model fails to correctly detect the plastic cup due to variations in appearance or partial visibility. Second, even when the cup is detected, small errors in the estimated position can cause the robot to miss the grasping point during the reaching phase.



**(a)** Frame_1 **(b)** Frame_2 **(c)** Frame_3 **(d)** Frame_4

**(e)** Frame_5 **(f)** Frame_6 **(g)** Frame_7 **(h)** Frame_8

**Figure 4.5:** Frames from robot starting point to object position

Overall, these results demonstrate that the proposed perception and manipulation pipeline is capable of performing reliable pick-and-place operations across different regions of the workspace and with different object types. The experiments also highlight the influence of both object geometry and perception accuracy on the final manipulation performance.

## 4.3 Discussion

The experimental results demonstrate the effectiveness of the proposed vision-guided robotic manipulation system. The evaluation was performed at three levels: object detection performance, hybrid perception robustness, and the overall pick-and-place capability of the robotic system.

The comparison of different YOLOv8 models shows that YOLOv8s provides the best balance between detection accuracy and computational efficiency. Although YOLOv8l is significantly larger, it does not provide noticeable improvements in detection performance. This behavior is likely related to the relatively small size of the training dataset (about 1000 images), which limits the advantage of larger models.

The hybrid perception strategy further improves system robustness by combining RGB-based detection with depth verification. While the YOLO detector recognizes trained object classes, it may fail when objects are partially visible or when other objects appear in the workspace. By using depth information to detect height changes in the delivery zone, the system can identify whether the zone is occupied even when visual detection fails.

Finally, the real-world pick-and-place experiments demonstrate that the integrated system can reliably detect objects, estimate their positions, and perform manipulation tasks in different workspace regions. The results also show that object geometry affects grasping success. For example, the martini cup achieved the highest success rate due to its stable base, while the plastic cup occasionally caused detection or

reaching errors.

Overall, the experiments confirm that combining deep learning-based perception with depth verification enables reliable robotic manipulation in practical workspace conditions.

# Chapter 5

# Conclusion and Future Work

This thesis presented the development and implementation of a vision-guided robotic system for cup detection and pick-and-place tasks in the beverage industry. Unlike many research systems that remain limited to laboratory environments, the proposed solution was implemented and tested within a real industrial setting, demonstrating the integration of perception and robotic manipulation under practical working conditions.

The experimental results show that the system can reliably perform both perception and manipulation tasks in an operational environment. One of the main contributions of this work is the successful transition from a research prototype to a system designed with industrial deployment in mind. Following the research and development phase, the system is planned to be integrated into the next generation of the company's robotic platform, where it will undergo further testing during real working cycles.

Despite the promising performance, several limitations remain. In particular, transparent cups remain challenging for the perception system, as RGB-D sensors typically struggle with transparent materials. In addition, the performance of the detection model can be affected by variations in lighting conditions within the workspace.

Overall, the results indicate that integrating deep learning–based perception with robotic manipulation can provide a practical solution for real industrial applications beyond controlled laboratory environments.

### Future Work

Although the proposed system demonstrates strong performance in real production conditions, several directions remain for further improvement and extension.

First, future work could investigate multi-object detection and simultaneous manipulation, enabling the system to handle multiple cups or objects within the workspace and improve throughput in production scenarios.

Second, the current system relies on a fixed Eye-to-Hand camera configuration, which may limit flexibility and sensitivity to calibration errors. Future research could

explore Eye-in-Hand camera setups combined with visual servoing, allowing the robot to dynamically adjust its motion based on real-time visual feedback and improving manipulation accuracy.

Another potential direction is the expansion of the training dataset and the evaluation of more advanced perception models. Larger datasets and improved training strategies could further enhance detection robustness, particularly under challenging lighting conditions, occlusions, or reflective objects.

Finally, future studies could extend the system toward more complex manipulation tasks, such as grasp planning for objects with varying shapes, adaptive grasp strategies, and integration with full robotic beverage preparation workflows.

These directions would further improve the scalability, robustness, and industrial applicability of vision-guided robotic manipulation systems.

# Bibliography


[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012) (cit. on pp. 1, 11).

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587 (cit. on pp. 1, 11, 13).

[3] Shaoqing Ren, Kaiming He, and Ross Girshick. "Faster R-CNN: Towards Real-Time Object Detection". In: *NIPS*. 2015 (cit. on pp. 1, 11, 13).

[4] Joseph Redmon. "You Only Look Once: Unified Real-Time Object Detection". In: *CVPR*. 2016 (cit. on pp. 1, 11, 14).

[5] Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi. "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications". In: *Mechatronics* 55 (2018), pp. 248–266 (cit. on p. 1).

[6] Danica Kragic, Henrik I Christensen, et al. "Survey on visual servoing for manipulation". In: *Computational Vision and Active Perception Laboratory, Fiskartorpsv* 15 (2002), p. 2002 (cit. on p. 1).

[7] SICK AG. *WLG16P-1H162120A00 Photoelectric Sensor Datasheet*. 2024. URL: https://www.sick.com/media/pdf/8/48/748/dataSheet_WLG16P-1H162120A00_1218948_en.pdf (cit. on p. 3).

[8] Yichao Mao, Qi Lu, and Qing Xu. "Visual servoing control based on EGM interface of an ABB robot". In: *2018 Chinese Automation Congress (CAC)*. IEEE. 2018, pp. 3260–3264 (cit. on p. 5).

[9] ABB Robotics. *Application Manual: Externally Guided Motion (RobotWare 7)*. 3HAC073318-001. ABB AB. 2022. URL: https://library.e.abb.com/public/a29a16d71dc244e49167d9ab089c7e14/3HAC073318%20AM%20Externally%20Guided%20Motion%20RW7-en.pdf (cit. on pp. 5, 19).

[10] MakrShakr. *MakrShakr website*. URL: https://www.makrshakr.com (cit. on p. 6).

[11] Andrea Bonci, Pangcheng David Cen Cheng, Marina Indri, Giacomo Nabissi, and Fiorella Sibona. "Human-robot perception in industrial environments: A survey". In: *Sensors* 21.5 (2021), p. 1571 (cit. on p. 10).

[12] Christopher E Smith and Nikolaos P Papanikolopoulos. "Vision-guided robotic grasping: Issues and experiments". In: *Proceedings of IEEE International Conference on Robotics and Automation*. Vol. 4. IEEE. 1996, pp. 3203–3208 (cit. on p. 10).

[13] Cristiano Premebida, Rares Ambrus, and Zoltan-Csaba Marton. "Intelligent robotic perception systems". In: *Applications of mobile robots* (2018), pp. 111–127 (cit. on p. 10).

[14] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object detection in 20 years: A survey". In: *Proceedings of the IEEE* 111.3 (2023), pp. 257–276 (cit. on pp. 10, 12).

[15] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. "Deep learning for generic object detection: A survey". In: *International journal of computer vision* 128.2 (2020), pp. 261–318 (cit. on p. 10).

[16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016 (cit. on p. 10).

[17] Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. Vol. 1. Ieee. 2001, pp. I–I (cit. on p. 11).

[18] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. Ieee. 2005, pp. 886–893 (cit. on p. 11).

[19] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448 (cit. on pp. 11, 13).

[20] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37 (cit. on pp. 11, 14).

[21] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. "Object detection with deep learning: A review". In: *IEEE transactions on neural networks and learning systems* 30.11 (2019), pp. 3212–3232 (cit. on pp. 11, 27).

[22] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. "End-to-End Object Detection with Transformers". In: *ECCV*. 2020 (cit. on pp. 12, 15).

[23] Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. "Yolo-world: Real-time open-vocabulary object detection". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2024, pp. 16901–16911 (cit. on pp. 12, 16).

[24] Ramona Kühlechner. “Object detection survey for industrial applications with focus on quality control”. In: *Production Engineering* 19.6 (2025), pp. 1271–1291 (cit. on pp. 13–15).

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Spatial pyramid pooling in deep convolutional networks for visual recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916 (cit. on p. 13).

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. “Feature pyramid networks for object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125 (cit. on p. 13).

[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988 (cit. on p. 15).

[28] Hei Law and Jia Deng. “Cornernet: Detecting objects as paired keypoints”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 734–750 (cit. on p. 15).

[29] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. “Centernet: Keypoint triplets for object detection”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6569–6578 (cit. on p. 15).

[30] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. “Deformable detr: Deformable transformers for end-to-end object detection”. In: *arXiv preprint arXiv:2010.04159* (2020) (cit. on p. 15).

[31] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. “Detrs beat yolos on real-time object detection”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2024, pp. 16965–16974 (cit. on p. 16).

[32] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *International conference on machine learning*. PmLR. 2021, pp. 8748–8763 (cit. on p. 16).

[33] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. “Open-vocabulary object detection via vision and language knowledge distillation”. In: *arXiv preprint arXiv:2104.13921* (2021) (cit. on p. 16).

[34] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. “Detecting twenty-thousand classes using image-level supervision”. In: *European conference on computer vision*. Springer. 2022, pp. 350–368 (cit. on p. 16).

[35] Liunian Harold Li et al. “Grounded language-image pre-training”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10965–10975 (cit. on p. 16).

[36] Shilong Liu et al. “Grounding dino: Marrying dino with grounded pre-training for open-set object detection”. In: *European conference on computer vision*. Springer. 2024, pp. 38–55 (cit. on p. 16).

[37] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017 (cit. on p. 16).

[38] Sicheng Wang, Milutin N Nikolić, Tin Lun Lam, Qing Gao, Runwei Ding, and Tianwei Zhang. “Robot manipulation based on embodied visual perception: A survey”. In: *CAAI Transactions on Intelligence Technology* 10.4 (2025), pp. 945–958 (cit. on p. 17).

[39] Md Tanzil Shahria, Md Samiul Haque Sunny, Md Ishrak Islam Zarif, Jawhar Ghommam, Sheikh Iqbal Ahamed, and Mohammad H Rahman. “A comprehensive review of vision-based robotic applications: Current state, components, approaches, barriers, and potential solutions”. In: *Robotics* 11.6 (2022), p. 139 (cit. on p. 17).

[40] Danica Kragic and Markus Vincze. *Vision for robotics*. Vol. 1. 1. Now Publishers Inc, 2009 (cit. on p. 17).

[41] Luis Pérez, Íñigo Rodríguez, Nuria Rodríguez, Rubén Usamentiaga, and Daniel F García. “Robot guidance using machine vision techniques in industrial environments: A comparative review”. In: *Sensors* 16.3 (2016), p. 335 (cit. on p. 17).

[42] Ya-Ling Chen, Yan-Rou Cai, and Ming-Yang Cheng. “Vision-based robotic object grasping—a deep reinforcement learning approach”. In: *Machines* 11.2 (2023), p. 275 (cit. on p. 17).

[43] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection”. In: *The International journal of robotics research* 37.4-5 (2018), pp. 421–436 (cit. on p. 17).

[44] Abhilasha Singh, V Kalaichelvi, and Ramanujam Karthikeyan. “A survey on vision guided robotic systems with intelligent control strategies for autonomous tasks”. In: *Cogent Engineering* 9.1 (2022), p. 2050020 (cit. on pp. 17, 18).

[45] Zhihua Qu, John Dorsey, Darren M Dawson, and Roger W Johnson. “Linear learning control of robot motion”. In: *Journal of Robotic systems* 10.1 (1993), pp. 123–140 (cit. on p. 18).

[46] Spiros G Papaioannou. “Interpolation algorithms for numerical control”. In: *Computers in industry* 1.1 (1979), pp. 27–40 (cit. on p. 18).

[47] Koichi Hashimoto. “A review on vision-based control of robot manipulators.” In: *Advanced Robotics* 17.10 (2003) (cit. on p. 18).

[48] Ikenna Enebuse, Mathias Foo, Babul Salam Ksm Kader Ibrahim, Hafiz Ahmed, Fhon Supmak, and Odongo Steven Eyobu. "A comparative review of hand-eye calibration techniques for vision guided robots". In: *IEEE Access* 9 (2021), pp. 113143–113155 (cit. on p. 18).

[49] Frank C Park and Bryan J Martin. "Robot sensor calibration: solving AX= XB on the Euclidean group". In: *IEEE Transactions on Robotics and Automation* 10.5 (1994), pp. 717–721 (cit. on p. 18).

[50] Roboception GmbH. *Hand-Eye Calibration*. `https://doc.rc-visard.com/latest/en/handeye_calibration.html`. 2024 (cit. on p. 18).

[51] Klaus H Strobl and Gerd Hirzinger. "Optimal hand-eye calibration". In: *2006 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2006, pp. 4647–4653 (cit. on p. 18).

[52] Kwang-Hee Lee, Hyun-Su Kim, Seung-Joon Lee, Sung-Won Choo, Sang-Moo Lee, and Kyung-Tae Nam. "High precision hand-eye self-calibration for industrial robots". In: *2018 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE. 2018, pp. 1–2 (cit. on p. 18).

[53] Justinas Miseikis, Kyrre Glette, Ole Jakob Elle, and Jim Torresen. "Automatic calibration of a robot manipulator and multi 3d camera system". In: *2016 IEEE/SICE International Symposium on System Integration (SII)*. IEEE. 2016, pp. 735–741 (cit. on p. 18).

[54] Jon Postel. *User datagram protocol*. Tech. rep. 1980 (cit. on p. 20).

[55] Srđan Popić, Dražen Pezer, Bojan Mrazovac, and Nikola Teslić. "Performance evaluation of using Protocol Buffers in the Internet of Things communication". In: *2016 international conference on smart systems and technologies (SST)*. IEEE. 2016, pp. 261–265 (cit. on p. 20).

[56] ABB Motion Program Execution. *EGM Support Documentation*. `https://abb-motion-program-exec.readthedocs.io/en/stable/egm.html`. Accessed: 2025. 2024 (cit. on p. 20).

[57] Roman Parak. *ABB EGM Python Library*. `https://github.com/rparak/ABB_EGM_Python`. 2023 (cit. on p. 21).

[58] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe. 2009, p. 5 (cit. on p. 21).

[59] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. "Robot operating system 2: Design, architecture, and uses in the wild". In: *Science robotics* 7.66 (2022), eabm6074 (cit. on p. 21).

[60] Yuya Maruyama, Shinpei Kato, and Takuya Azumi. "Exploring the performance of ROS2". In: *Proceedings of the 13th international conference on embedded software*. 2016, pp. 1–10 (cit. on p. 21).

[61] Florian Lobert. *ABB-EGM-Python: Python Interface for ABB Externally Guided Motion*. `https://github.com/FLo-ABB/ABB-EGM-Python`. GitHub repository, Accessed: 2025. 2023 (cit. on p. 33).